Open Source Product

이 문서는 오픈 소스 (Open Source) 솔루션에 대한 다양한 정보를 공유하기 위해 작성되었습니다.

2024-03-07 08:57:02.167

Selenium 가이드

이 문서는 Selenium 을 이용한 웹 테스트 자동화 방법을 정리한다. 개요 Selenium은 웹 브라우저를 자동화하기 위한 강력하고 널리 사용되는 오픈 소스 프레임워크입니다. 웹 페이지와 상호 작용하기 위한 테스트 스크립트를 작성하고, 사용자 작업을 시뮬레이션하며, 다양한 브라우저와 플랫폼에서 웹 애플리케이션의 동작을 검증할 수 있는 도구와 라이브러리를 제공한다. Selenium은 Java, C#, Python, Ruby, JavaScript를 포함한 다양한 프로그래밍 언어를 지원하여 웹 애플리케이션 테스팅에 있어 다재다능한 선택지를 제공한다. 제공 기능 다양한 언어 지원: Selenium은 Java, C#, Python, Ruby, JavaScript 등 여러 프로그래밍 언어를 지원하여 다양한 개발 환경에서의 사용을 가능하게 한다. 크로스 브라우저 테스팅: Selenium WebDriver를 통해 Chrome, Firefox, Internet Explorer, Safari 등 다양한 웹 브라우저에서 웹 애플리케이션을 자동으로 테스트할 수 있다. 웹 애플리케이션 자동화: 사용자의 웹 브라우징 활동을 모방하여 웹 페이지와의 상호 작용을 자동화할 수 있습니다. 이는 폼 제출, 마우스 클릭, 데이터 입력 및 추출 등을 포함한다. Selenium IDE: 브라우저 확장 프로그램으로 제공되며, 코드 작성 없이 테스트 케이스를 기록하고 재생할 수 있는 기능을 제공합니다. 이를 통해 비개발자도 쉽게 테스트 스크립트를 생성할 수 있다. Selenium Grid: 여러 시스템과 브라우저에서 동시에 테스트를 실행할 수 있게 해주어, 대규모 테스트 실행과 분산 테스팅이 가능합니다. 이는 테스트의 실행 시간을 단축시키고 효율성을 높여 준다. 오픈 소스: Selenium은 오픈 소스 소프트웨어로, 무료로 사용할 수 있으며, 활발한 커뮤니티의 지원을 받아 지속적으로 개선되고 있다. Selenium 테스트 자동화 구성 요소 Selenium은 여러 컴포넌트로 구성됩니다: Selenium WebDriver: 직접 브라우저 자동화와 상호 작용을 허용하는 핵심 컴포넌트입니다. 브라우저에 명령을 보내고 결과를 검색하여 웹 애플리케이션의 자동화를 가능하게 한다. Selenium IDE (Integrated Development Environment): 코드 작성 없이 상호 작용을 녹화 및 재생할 수 있는 Chrome과 Firefox 확장 프로그램으로, 테스트 스크립트를 빠르게 생성하는 데 유용하다. Selenium Grid: 다양한 머신과 브라우저에서 동시에 Selenium 테스트를 실행할 수 있게 하여 분산 테스트 실행을 가능하게 한다. https://browserstack.wpenginepowered.com/wp-content/uploads/2021/09/Selenium-WebDriver-for-Automation-Testing.webp Java Language 예시 사전 준비 사항 Java Development Kit (JDK): 시스템에 JDK가 설치되어 있는지 확인하세요. 공식 Oracle 웹사이트 또는 다른 JDK 제공업체에서 다운로드할 수 있다. Integrated Development Environment (IDE): Eclipse, IntelliJ IDEA, NetBeans와 같은 IDE입니다. 이 가이드는 Eclipse를 사용한다고 가정한다. Maven: 프로젝트의 빌드와 의존성을 관리하는 프로젝트 관리 도구입니다. Eclipse와 IntelliJ IDEA에 통합되어 있다. WebDriver: 테스트하고자 하는 브라우저용 WebDriver 실행 파일을 다운로드해야 한다(예: Google Chrome용 ChromeDriver, Firefox용 geckodriver). Selenium 테스트 프로젝트 생성 단계 Maven 프로젝트 생성: Eclipse를 열고 파일 > 새로 만들기 > Maven 프로젝트를 선택한다. 새 Maven 프로젝트를 생성하기 위한 마법사를 따릅니다. 기본 아키타입을 사용할 수 있다. Selenium 의존성 추가: pom.xml 파일을 편집하여 Selenium 의존성을 포함시킵니다. <dependencies> 섹션 내에 다음을 추가한다: LATEST_VERSION을 사용하고자 하는 최신 Selenium 버전으로 교체한다. <dependency> <groupId>org.seleniumhq.selenium</groupId> <artifactId>selenium-java</artifactId> <version>LATEST_VERSION</version> </dependency> 스트 클래스 작성: Java를 사용하여 웹 페이지를 열고, 제목을 검증하는 간단한 테스트 케이스를 작성합니다. import org.openqa.selenium.WebDriver; import org.openqa.selenium.remote.DesiredCapabilities; import org.openqa.selenium.remote.RemoteWebDriver; import org.junit.Assert; import org.junit.After; import org.junit.Before; import org.junit.Test; import java.net.URL; public class GridTitleTest { private WebDriver driver; @Before public void setUp() throws Exception { // 원하는 브라우저 구성을 설정합니다. DesiredCapabilities capabilities = DesiredCapabilities.chrome(); // 여기에서 chrome을 다른 브라우저로 변경할 수 있습니다. // Selenium Grid Hub URL을 설정합니다. 이 URL은 Selenium Grid Hub에 대한 실제 URL로 변경해야 합니다. URL hubUrl = new URL("http://localhost:4444/wd/hub"); // 원격 WebDriver 인스턴스를 생성합니다. driver = new RemoteWebDriver(hubUrl, capabilities); } @Test public void testWebTitle() { // 테스트할 웹 페이지를 엽니다. driver.get("http://www.example.com"); // 페이지 제목을 가져와서 검증합니다. String pageTitle = driver.getTitle(); Assert.assertEquals("Title of this web page", pageTitle); } @After public void tearDown() { // 테스트 후에 브라우저를 닫습니다. if (driver != null) { driver.quit(); } } } 설정(Setup): setUp 메서드에서는 원격 WebDriver 인스턴스를 생성합니다. 이때, Selenium Grid Hub의 URL과 원하는 브라우저 구성(예: Chrome)을 지정한다. 테스트 실행(Test): testWebTitle 메서드에서는 웹 페이지를 열고, 그 제목이 "Title of this web page"와 일치하는지 검증한다. 정리(Teardown): tearDown 메서드에서는 테스트가 완료된 후 브라우저를 닫는다. WebDriver 설정: 테스트 코드에서 시스템 속성을 설정하여 다운로드한 WebDriver 실행 파일의 경로를 지정한다. 테스트 실행: Eclipse에서 작성한 테스트를 실행하고, 웹 페이지가 예상대로 동작하는지 확인한다. UI 요소 찾는 방법 Selenium은 다양한 방식으로 UI 구성 요소를 접근하는 방법을 제공한다. Id Links Tag names CSS Selectors XPath JQuery Text 예시) driver.findElement(By.id(<elementID>))

2014 - 2023 CURVC Corp. All rights reserved

driver.findElement(By.linkText(<linktext>)) Id driver.findElement(By.id (“name of id”)) UI 요소를 찾는 가장 이상적인 솔루션 ID가 존재하지 않을 수 있음 복수의 Id가 존재 할 수 있음 Links driver.findElement(By.linkText(“name of link”)) driver.findElement(By.partialLinkText(“name of link”)) 링크에 표시된 텍스트로 요소 찾기 Tag driver.findElement(By.tagName(“name of HTML tag”)) 태그 이름 HTML 태그를 기반으로 웹 요소 찾기 CSS Selectors Absolute path: driver.findElement(By.cssSelector(“html>body>div>p>input”)) Relative path: driver.findElement(By.cssSelector(“input”)) *first instance found Regula attribut: driver.findElement(By.cssSelector(“button[name='button name']”)) Special attributes: id: driver.findElement(By.cssSelector(“#save")); tag & id: driver.findElement(By.cssSelector(“button#save")); class attribute: driver. findElement(By.cssSelector(“.yoyo")); tag & class attribute: river.findElement (By.cssSelector(“input.username")); tag with attribute value: driver. findElement(By.cssSelector(“img[alt=kuku"]")); tag which has attribute: driver.findElement(By.cssSelector(“img[alt]")); tag which doesn"t have attribute: driver.findElement(By.cssSelector(“img:not([alt])")); XPath /html /body/div[5]/div[2]/div/div[2]/div[2]/h2[1] .//*[@id='answers']/h2[1]/a[1] XPath는 XML에서 노드를 찾는 데 사용되는 방법임 찾으려는 요소에 적합한 ID 또는 이름 속성이 없는 경우에 적합 절대 XPath 보다 상대 XPath 사용 권장 XPath-Attributes는 ID나 이름을 사용할 수 없을 때 사용 권장 Syntax: htmltag [@attribute1='value1' and @attribute2='value2'] Example: input@id= 'Password' and @placeholder='Password'] WebDriver API 목록 driver. get ("http://www.google.com"); To open an application driver.findElement(By. id("passwd-id")); Finding Element using Id driver.findElement(By.name ("passwd")); Finding Element using Name driver.findElement(By. Xpath(" //input[@id='passwd-id']")); Finding Element using Xpath element.sendKeys ("some text"); To type some data lement. clear(); clear the contents of a text eld or text area driver.findElement(By. Xpath("//select")); Selecting the value findElement(By.id("submit")).click(); To click on Any button/Link driver. switchTo().window("window Name"); Moving from one window to another window driver.switchTo().frame("frame Name"); swing from frame to frame (or into iframes) driver.get("http://www.google.com"); To open an application driver.switchTo().frame("frameName.0.child"); to access sub frames by separating the path with a dot, and you can specify the frame by its index too driver.switchTo().alert(); Handling Alerts river.Navigate().to ("http://www.example.com"); To Navigate Particular URL driver.Navigate(). forward(); To Navigate Forward river.Navigate().back(); To Navigate Backward driver.close(); Closes the current window driver.Quit(); Quits the driver and closes every associated window driver.switch_to_alert(); Switches focus to an alert on the page. driver.Refresh(); Refreshes the current page driver.get_screenshot_as_file('/Screenshots/foo.png'); The full path you wish to save your screenshot to driver.get_screenshot_as_base64(); Gets the screenshot of the current window as a base64 encoded string select. findElements(By.tagName("option")); Selecting the value select.deselectAll(); This will deselect all OPTIONs from the rst SELECT on the page select. selectByVisibleText("Edam"); select the OPTION with the displayed text of “Edam”

2024-02-06 08:14:00.451

Eclipse

이 문서는Eclipse 사용에 대한 가이드를 공유하기 위해 작성되었다.

2018-03-09 20:31:11.076

Redmine 가이드

2014 - 2023 CURVC Corp. All rights reserved

2020-04-22 12:26:15.27

Git 가이드

이 문서는 Git 가이드를 공유하기 위해 작성되었다. image2019-6-15_13-14-24.png image2019-6-15_13-15-58.png image2019-6-15_13-16-16.png image2019-6-15_13-16-35.png image2019-6-15_13-16-54.png image2019-6-15_13-17-10.png 1. Git 시작하기 Git 저장소 https://confluence.curvc.com/pages/viewpage.action?pageId=51578247&src=contextnavpagetreemode git init git clone git config 변경 저장하기 git add git commit git diff git stash .gitignore 저장소 점검하기 git status git log git tag git blame 변경 취소하기 git 실행 취소 git clean git revert git reset git rm Rewriting history https://confluence.curvc.com/display/ASD/Rewriting+history?src=contextnavpagetreemode git commit --amend https://confluence.curvc.com/display/ASD/git+commit+--amend?src=contextnavpagetreemode git rebase https://confluence.curvc.com/display/ASD/git+rebase?src=contextnavpagetreemode git reflog https://confluence.curvc.com/display/ASD/git+reflog?src=contextnavpagetreemode 2. Git 협업하기 동기화하기 git remote https://confluence.curvc.com/display/ASD/git+remote?src=contextnavpagetreemode git fetch https://confluence.curvc.com/display/ASD/git+fetch?src=contextnavpagetreemode git pull https://confluence.curvc.com/display/ASD/git+pull?src=contextnavpagetreemode git push https://confluence.curvc.com/display/ASD/git+push?src=contextnavpagetreemode 브랜치 사용하기 git branch https://confluence.curvc.com/display/ASD/git+branch?src=contextnavpagetreemode git checkout https://confluence.curvc.com/display/ASD/git+checkout?src=contextnavpagetreemode git merge https://confluence.curvc.com/display/ASD/git+merge?src=contextnavpagetreemode 병합 충돌 해결하기 (Merge conflicts) https://confluence.curvc.com/pages/viewpage.action?pageId=51578393&src=contextnavpagetreemode 병합 전략 (Merge strategies) Pull request 만들기

2018-09-19 11:41:39.967

XWiki 소개

이 페이지는 xWiki의 간단한 화면 덤프를 공유하기 위해 작성되었다. Wiki Home Wiki Home은 별도의 페이지이고 아래와 같이 구성할 수 있다. image2013-10-11 11:5:19.png Space 생성 문서를 정리하기 위해 먼저 Space(Project)를 생성한다. image2013-10-11 11:7:43.png 스페이스 대시보드 스페이스 대시보드는 페이지로 구성된다. image2013-10-11 11:12:52.png 페이지 생성 스페이스 내부에 페이지를 생성하여 문서를 작성한다. image2013-10-11 11:8:45.png 페이지 작업 아래는 Wiki 페이지 작성을 위한 에디터의 예시를 보여준다. image2013-10-11 11:11:35.png 생성된 페이지 다음은 생성된 페이지의 예시를 보여준다. image2013-10-11 11:12:7.png

2018-03-09 20:33:22.775

JMeter 가이드

2014 - 2023 CURVC Corp. All rights reserved

이 문서는 JMeter 가이드를 공유하기 위해 작성되었따. Apache JMeter는 오픈 소스 소프트웨어로 기능 행위의 로드 테스트와 성능 측정을 위해 설계된 순수 Java 어플리케이션입니다. JMeter는 웹어플리케이션을 위해 설계되었지만 다른 테스트 기능으로 확장할 수 있습니다. Apache JMeter는 정적 및 동적 자원, 웹 동적 어플리케이션 모두에서 성능을 테스트하는 데 사용될 수 있습니다. 서버, 서버 그룹, 네트워크 또는 개체의 과부하를 시뮬레이션하여 강도를 테스트하거나 다양한 로드 유형에서 전반적인 성능을 분석하는데 사용할 수 있습니다. Apache JMeter 기능은 다음과 같다. 다양한 어플리케이션/서버/프로토콜 타입을 로드하고 성능 테스트할 수 있는 기능 Web - HTTP, HTTPS(Java, NodeJS, PHP, ASP.NET...) SOAP / REST Webservices FTP JDBC를 통한 데이터베이스 LDAP JMS를 통한 메시지 기반 미들웨어 메일 - SMTP(S), POP3(S), IMAP(S) 네이티브 명령 혹은 쉘 스크립트 TCP Java 오브젝트 빠른 테스트 플랜 레코딩(브라우저 혹은 네이티브 어플리케이션), 빌드, 디버깅이 지원되는 완전한 기능의 테스트 IDE 모든 Java 호환 OS(Linux, Windows, Mac OSX 등)에서 테스트를 로드하는 코멘드 라인 모드 완전하고 준비된 동적 HTML 보고서 HTML, JSON, XML 혹은 어떤 텍스트 포멧, 가장 인기있는 응답 포멧으로부터 데이터 추출 능력 풀 멀티 쓰레딩 프레임워크 많은 쓰리드로부터 병렬 샘플링 분리된 쓰레드 그룹으로 다른 기능의 동시 샘플링 캐싱과 테스트 결과의 오프라인 분석/반복수행 고도로 확장 가능한 코어 플러그인 샘플러는 무제한 테스팅 수용력을 허용 Groovy와 BeanShell과 같은 스크립트가능한 샘플러 플러그인 터이머내에 선택가능한 로드 통계 데이터 분석과 비주얼라이제이션 플러그인 동적 입력을 제공하는데 사용되는 기능 쉬운 지속적인 통합 라이브러리

2022-04-13 13:33:07.948

CentOS 가이드

이 문서는 CentOS Linux 관련 가이드를 공유하기 위해 작성되었다.

2021-03-02 17:05:18.64

GitLab 가이드

2020-10-20 15:14:23.849

Nginx 가이드

이 문서는 Nginx 가이드를 공유하기 위해 작성되었다.

2018-02-05 14:20:20.341

Open Source Knowledge Base

2014 - 2023 CURVC Corp. All rights reserved

2020-09-10 10:12:22.557

Open Source 설치가이드

이 문서는 개발환경과 관련된 가이드를 공유하기 위해 작성되었다.

2021-08-24 13:33:46.288

Apache Httpd

이 문서는 Apache httpd 웹 서버에 대한 가이드를 공유하기 위해 작성되었다.

2018-03-09 20:34:25.242

Subversion 가이드

이 문서는 Subversion 가이드를 공유하기 위해 작성되었다.

2018-03-09 20:32:54.01

Jenkins 가이드

이 문서는 Jenkins 가이드를 공유하기 위해 작성되었다.

2018-03-09 20:30:37.741

Docker 가이드

이 문서는 Docker 사용에 대한 가이드를 공유하기 위해 작성되었다.

2022-04-14 15:00:22.132

쿠버네티스(Kubernetes) 가이드

이 문서는 쿠버네티스(Kubernetes - k8s) 가이드를 공유하기 위해 작성되었다.

2014 - 2023 CURVC Corp. All rights reserved

2018-03-09 20:31:26.865

Gerrit 가이드

2018-03-09 22:25:28.844

Eclipse Che 가이드

이 문서는 eclipse che 가이드를 공유하기 위해 작성되었다.

2020-05-20 16:19:01.373

Gradle

2018-03-09 20:33:49.372

LDAP 가이드

이 문서는 LDAP 가이드를 공유하기 위해 작성되었다.

2024-02-06 08:35:38.181

Command 실행 메뉴 추가하기

이 문서는 Eclipse 에 Windows batch 스크립트 실행 메뉴를 추가하는 방법을 정리한다. 배경 반복적으로 Windows batch 스크립트 실행이 필요한 경우 Eclipse에서 제공하는 External Tool 구성을 통해 실행 메뉴를 추가하여 실행할 수 있다. 구성 Run > External Tools > External Tools Configurations Screenshot 2024-02-05 at 11.09.31PM.png New Configuration Screenshot 2024-02-06 at 8.25.18AM.png 실행 내용 구성 Name: 표시할 메뉴 이름 설정 Main Tab Location: 실행 파일 (스크립트 포함) 경로 지정 Workspace 또는 특정 파일 시스템 경로 지정 가능 Working Directory: 실행 경로 지정 Workspace 또는 특정 파일 시스템 경로 지정 가능 Arguments: 실행 파일에 전달할 변수 지정 가능 Screenshot 2024-02-06 at 8.26.54AM.png Refresh Tab 실행 후 갱신할 범위를 지정한다. Screenshot 2024-02-05 at 11.10.34PM.png 실행 추가한 구성은 Run 메뉴에 노출된다. Screenshot 2024-02-06 at 8.33.12AM.png

2014 - 2023 CURVC Corp. All rights reserved

2017-02-22 20:37:01.764

Redmine 사용자 가이드

이 문서는 Redmine의 일반 사용자에 대한 가이드 제공한다. 1. Personal Information 1.1 My Account 나의 계정(My Account)에서는 계정의 기본정보, 이메일 알림 설정, 언어 등의 설정을 수행할 수 있다. 나의 계정으로 접속하기 위해서 상단 자신의 아이디 옆 My Account 계정을 선택한다. Information First name : 이름 Last name : 성 Email : 이메일 Language : 사용하고자 하는 언어 선택 Email notifications For any event on all my projects 나의 모든 프로젝트의 알림을 받음 For any event on the selected projects only... 선택된 프로젝트만 알림을 받음 Only for things I watch or I'm involved in 내가 포함되어 있거나 보고 있는 이슈 Only for things I am assigned to 내가 할당된 이슈 Only for things I am the owner of 내가 소유자인 이슈 No events 받지 않음 Preferences Hide my email address : 이메일 주소 숨김 Time zone : 표시를 원하는 시간 대역 Display comments : 코멘트 출력 Warn me when leaving a page with unsaved text : 저장 알림 image2017-2-21_16-33-10.png 1.2 My Page 마이 페이지(My Page)는 프로젝트에 대한 다양한 정보를 보여준다. 기본적으로 다음 두개의 정보 블록을 제공한다. Issue assigned to me 전체 프로젝트에서 현재 사용자에게 할당된 이슈 리스트를 보여준다. Reported issues 전체 프로젝트에서 현재 사용자가 보고 받는 모든 이슈 리스트를 보여준다. 상단 Personalize this page 버튼을 클릭하면, 개인이 운하는 블록을 추가할 수 있다. Watched Issues 현재 사용자가 watch하고 있는 이슈 리스트 제공 Latest news 최근 뉴스 overview 제공 Calendar Weekly Calendar overview 제공 Documents 최신 문서의 overview 제공 Spent time 현재 사용자의 최근 7일 동안의 전체 소비한 시간 제공 image2017-2-22_14-23-53.png 2. Project Overview 프로젝트 Overview 페이지는 프로젝트의 전체 Overview를 보여준다. Issue tracking 얼마나 많은 업무들이 Open되고 Close되어 있는지를 나타낸다. Member 해당 프로젝트의 멤버가 누군지를 나타낸다. Latest news 해당 프로젝트의 최신 뉴스를 보여준다. Progress 프로젝트 Roadmap의 각 버전별 진행상황을 보여준다. Spent time 프로젝트의 전체 작업 소비 시간을 보여준다. image2017-2-22_14-23-1.png 3. Issue Tracking 이슈는 Redmine 프로젝트를 구성하는 핵심 사항이다. 하나의 이슈는 프로젝트에 종속되며, 한명의 사용자에게 소유되고 하나의 버전과 연관된다. 3.1 Issue List 프로젝트 메뉴의 Issues를 선택하면 현재 프로젝트의 이슈 리스트를 확인할 수 있다. image2017-2-22_14-39-8.png 3.1.1 Filters Filters에서는 이슈를 다양한 조건으로 필터하여 리스트를 보여준다. image2017-2-22_14-42-44.png 3.1.2 Options Options은 이슈 리스트에 몇몇 옵션들을 추가하여 리스트를 보여준다. Columns : 이슈 리스트에 보여줄 컬럼 항목을 설정한다. Group results by : 설정한 항목에 따라 이슈를 그룹하여 보여준다. Show : 이슈 리스트에 Description을 포함한다. image2017-2-22_14-44-8.png 3.2 Issue 생성 이슈를 생성하기위해 프로젝트 메뉴에서 New issue 링크를 선택한다. 기본적으로 제공되는 Issue의 항목은 다음 표와 같다. 필드 필수여부 설명 Private 옵션 개인 이슈로 만든다. Tracker 필수 이슈의 타입을 선택한다. Subject 필수 이슈의 주제(제목)를 입력한다. Description 옵션 이슈의 세부적인 설명을 입력한다. Status 필수 이슈의 상태를 선택한다. Priority 필수 이슈의 중요도 혹은 우선순위를 선택한다. Assignee 옵션 이슈의 담당자를 선택한다. Target version 옵션 이슈가 적용될 버전을 선택한다. Parent task 옵션 만약 Sub Task일 경우 부모 Task를 선택한다. Start date 옵션 시작 날짜를 입력한다. Due date 옵션 종료 날짜를 입력한다. Estimated time 옵션 일의 시간을 추정한다. % Done 옵션 업무의 완료 정도를 %로 입력한다. Files 옵션 이슈와 관련된 파일을 업로드 한다. Watchers 옵션 이슈를 함께 볼 사용자를 선택한다. 다음은 이슈 생성화면의 예시를 보여주고 있다. image2017-2-22_20-31-1.png 이슈 생성 화면의 이슈 생성을 위한 버튼은 3가지가 존재한다. Create : 해당 이슈를 생성한다. Create and continue : 이슈 생성 후 다른 이슈를 다시 생성한다. Preview : 이슈의 Description에 대한 미리보기를 제공한다. 3.3 이슈 세부사항 이슈 리스트에서 해당 이슈를 선택하면, 이슈의 세부사항을 확인할 수 있다. image2017-2-22_15-9-0.png 이슈 화면의 상단 메뉴에서는 다음을 수행할 수 있다. Edit : 이슈 내용을 수정한다. Log time : 이슈에 대한 작업 시간을 기록한다. Watch : 해당 이슈를 Watch한다. 이슈에 대한 변경 사항이 발생하면 알림 메일을 받을 수 있다. Copy : 이슈를 복제하여 새로운 이슈를 만든다. Delete : 이슈를 삭제한다. 3.4 이슈 링크 레드마인에서는 이슈간의 관계를 다양한 형태로 정의하고 설정할 수 있다. 특정 이슈와의 링크가 설정된 이슈는 Related issues에 나타나며, 이슈의 링크 관계는 다음과 같이 표현될 수 있다. Related to - 관련있는 이슈를 연결 Duplicates - 중복되는 이슈를 연결 : 만약 B 이슈가 duplicates A이슈라면, A가 Close되면 B가 자동으로 Close된다. Duplicated by - 중복되는 이슈를 연결 : 만약 A 이슈가 ducplicated by B 이슈라면, A가 Close되면 B가 자동으로 Close된다. Blocks - A이슈가 block B이슈라면, A는 B가 Close되기전에 Close할 수 없다. Blocked by - B이슈가 blocked by A이슈라면, A는 B가 Close되기전에 Close할 수 없다. Precedes / Follows - 이슈의 우선순위 관계를 나타내는 링크 Copied from / to - 특정 이슈를 복사했을 경우 image2017-2-22_15-32-42.png 4. Activity Activity에서는 프로젝트에 일어나고 있는 전체 활동을 시간 순으로 나타낸다. image2017-2-22_15-50-46.png Activity의 범위는 좌측 Activity 메뉴에서 선택 후 적용하면 확인할 수 있다. 다음은 Activity에 추가할 수 있는 항목을 보여준다. Issues Changesets News Documents Files Wiki edits Messages Spent time 5. Roadmap 5.1 Roadmap List 레드마인의 로드맵은 프로젝트의 개발을 계획하고 관리하기 위해 버전 기반으로 이슈를 구성하는 기능을 제공한다. 현재 프로젝트의 로드맵을 확인하고 버전을 설정하기 위해 프로젝트 메뉴에서 Roadmap 메뉴를 선택한다. 다음은 Roadmap에 대한 예제 화면을 보여주고 있다. image2017-2-22_16-15-4.png 로드맵 페이지에는 각 버전에 대한 제목, 종료일, 설명, 진행상태, 이슈를 제공한다. 좌측 메뉴에서 Version에 대한 특정 이슈를 보거나 특정 버전만을 볼 수 있

2014 - 2023 CURVC Corp. All rights reserved

는 필터를 제공한다. 5.2 New Version 프로젝트에 새로운 버전을 추가하기 위해 Roadmap 화면 상단의 New version 링크를 클릭한다. 아래 그림과 같이 New version 페이지가 나타나면 다음을 참고하여 필요한 정보를 입력한다. Name : 버전의 이름으로 프로젝트 혹은 조직의 정책에 따라 명명한다. Description : 버전에 대한 간단한 설명을 입력한다. Status open : 버전이 열리 있는 상태 locked : 버전이 lock 되어 있는 상태 - 버전에 이슈를 할당할 수 없다. close : 버전이 종료된 상태 - 버전에 이슈를 할당할 수 없고 reopen 할 수 없다. Wiki page : 버전에서 참고할 수 있는 위키 페이지 Date : 버전 종료 예정일 Sharing Not shared : 버전을 공유하지 않는다. With subprojects : 버전을 하위 프로젝트와 공유한다. With project hierarchy : 버전을 hierarchy 관계에 있는 모든 프로젝트와 공유한다.(조상과 자식) With project tree : 프로젝트 트리 상의 모든 프로젝트와 공유한다.(조상, 자식, 조상의 자식) With all projects : 모든 프로젝트와 버전을 공유한다. image2017-2-22_20-32-13.png 6. Gantt 간트는 할당된 버전을 기반으로 이슈들의 시작일과 종료일을 간트차트에 보여준다. 간트차트의 일정을 수정하기 위해서는 각 이슈의 시작일과 종료일을 수정함으로써 수정할 수 있다. Filters에서는 Gantt에 보여지는 이슈의 리스트를 필터할 수 있으며, Options에서는 이슈의 Blocks과 Precedes와 Progress line을 설정할 수 있다. 간트차트 상단 메뉴를 통해 Gantt차트의 범위 설정과 간트차트를 확대 및 축소할 수 있다. image2017-2-22_16-45-3.png 7. Calendar Calendar는 현재 프로젝트의 월별 이슈의 Overview를 제공한다. 프로젝트의 Calendar에 접근하기 위해 프로젝트 메뉴에서 Calendar 메뉴를 선택한다. Calendar에서는 모든 이슈의 시작일과 종료일을 나타내며 모든 버전의 종료일을 표시한다. 다음 그림은 Calendar의 예시를 보여준다. image2017-2-22_16-54-33.png 8. News, Documents, Wiki, File 8.1 News News는 프로젝트에 관련된 문서를 게시판과 같은 형태로 출력할 수 있게 해주는 모듈이다. News로 접근하기 위해 프로젝트 메뉴에서 News 메뉴를 선택한다. 다음 그림은 News의 예시를 보여주고 있다. image2017-2-22_20-33-1.png News 페이지의 상단 메뉴를 통해 News를 추가하거나 자신을 Watch로 추가할 수 있다. Add news : 새로운 뉴스를 추가한다. Watch : 뉴스 페이지를 Watch 한다. 8.2. Documents Documents는 프로젝트의 사용자 문서 혹은 기술 문서와 같은 문서를 작성할 수 있는 모듈이다. Documents로 접근하기 위해 프로젝트 메뉴에서 Documents 메뉴를 선택한다. New document : 새로운 Document를 생성하기 위해 Document 페이지의 상단에서 New document 를 선택한다. Document category : Document category는 기본적으로 사용자 문서와 기술 문서 두 가지로 설정되어 있으며, 필요에 따라 관리자 페이지에서 enum 값을 수정하거나 추가할 수 있다. 8.3 Wiki 8.3.1 새로운 페이지 생성 레드마인 Wiki에서 새로운 페이지 생성하기 위해서는 생성하고자 하는 페이지에서 다음 link 마크업을 추가하면 된다. [[MyNewWikiPage]] 8.3.2 Wiki 툴바 레드마인 Wiki의 상단에는 다음 툴바들을 제공한다. Edit : 현재 페이지의 컨텐츠를 수정한다. Watch : 자신을 현재 페이지에 대한 Watcher로 추가한다. 이를 통해 변경사항에 대한 알림 메일을 받을 수 있다. Rename : 위키페이지의 이름을 변경하거나 이동할 수 있다. Lock : Lock을 걸고 페이지를 수정할 수 있다. Delete : 해당 위키 페이지를 삭제한다. History : 위키 페이지에 대한 수정 사항을 확인하고 비교할 수 있다. 다음 그림은 위키 페이지의 수정 히스토리에 대한 예제 화면을 보여준다. image2017-2-22_19-9-3.png 8.3.3 Sidebar 레드마인 Wiki의 사이드바는 다음 링크를 제공한다. Start page : 설정된 시작 페이지로 이동한다. Index by title : 위키 페이지를 이름 순으로 정렬하여 보여준다. Index by date : 위키 페이지를 작성일 순으로 정렬하여 보여준다. 8.3.4 Wiki Markdown 이번절에서는 레드마인에서 사용할 수 있는 기본적인 Wiki 마크업에 대해서 설명한다. Ckeditor 혹은 HTML을 지원하는 에디터의 경우에는 HTML 문법을 사용하여 Wiki 페이지를 작성할 수 있다. 텍스트 스타일은 다음과 같은 방식으로 적용할 수 있다. = Heading 1 = == Heading 2 == === Heading 3 === ==== Heading 4 ==== ===== Heading 5 ===== ====== Heading 6 ====== 불릿 리스트는 다음과 같은 방식으로 적용할 수 있다. * Item1 * Item2 * Item3 * Item4 ** Sub-item 4 a) *** Sub-item 4 a) 1. **** Sub-item 4 a) 1. i) **** Sub-item 4 a) 1. ii) ** Sub-item 4 b) * Item5 넘버 리스트는 다음과 같은 방식으로 저용할 수 있다. # Item1 # Item2 # Item3 # Item4 ## Sub-item 1 ### Sub-sub-item #### Sub-sub-sub-item ## Sub-item 2 # Item5 리치 텍스트는 다음과 같은 방식으로 적용할 수 있다. ''italicize text'' '''bold the text''' 레드마인의 Markdown에 대한 자세한 정보는 다음 링크를 참조한다. http://www.redmine.org/projects/redmine/wiki/RedmineTextFormattingMarkdown http://www.redmine.org/projects/redmine/wiki/RedmineTextFormattingMarkdown 8.4 Files 레드마인의 Files 모듈에서는 프로젝트에 필요한 파일들을 첨부하고 공유할 수 있다. Files 모듈의 파일 리스트는 다음 정보를 함께 제공한다. File - 파일명 Date - 업로드 날짜 Size - 파일 사이즈 D/L - 다운로드 수 MD5 - MD5 해시 정보 9. Repository 레드마인의 Repository 모듈에서는 프로젝트의 소스 코드 저장소와 최신 Commit을 확인할 수 있게 해준다. 9.1 코드 탐색 디렉토리의 + 버튼을 통해 확장하여 브라우징할 수 있으며, 디렉토리를 더블 클릭하거나 Enter 버튼을 클릭하여 디렉토리로 진입할 수 있다. image2017-2-22_20-33-48.png 9.2 Revision 하단 Latest revisions에서는 변경 히스토리를 확인할 수 있고 각 revision 버전을 선택하여 세부사항을 확인할 수 있다. revision 간의 차이점은 비교를 원하는 revision을 선택하고 View differences 버튼을 클릭하여 비교할 수 있다. 다음은 리비전의 차이점을 side by side view로 보여주는 예시이다. image2017-2-22_20-34-36.png 9.3 Statistics Repository 페이지 상단의 Statistics를 선택하여 월간 Commit 추이와 사용자별 Commit 수에 대한 차트를 확인할 수 있다. 10. Project Settings 10.1 Information 프로젝트 세팅의 Information 탭은 프로젝트의 대한 일반적인 사항에 대한 설정을 제공한다. Name : 프로젝트 이름을 설정한다. Description : 프로젝트에 대한 설명을 설정한다. Identifier : 프로젝트 식별자를 확인한다. Homepage : 관련 홈페이지의 링크를 설정한다. Public : 프로젝트를 공용으로 설정한다. Subproject of : 서브프로젝트로 설정한다. Inherit members : 사용자를 상속 받는다. Trackers : 사용할 이슈 타입을 선택한다. image2017-2-22_20-35-58.png 10.2 Modules Modules 탭에서는 현재 프로젝트에서 사용할 모듈들을 선택할 수 있게 해준다. 모듈을 플러그인 설치 여부에 따라 다르게 나타날 수 있다. 10.3 Members Members 탭에서는 현재 프로젝트의 멤버와 역할을 추가/수정/삭제 할 수 있다. 다음 그림은 Members 탭의 예제 그림을 보여준다. image2017-2-22_19-40-8.png 10.4 Versions 버전 탭에서는 현재 프로젝트에서 사용되고 있는 전체 버전의 정보를 보여준다. 리스트에서 함께 제공되

2014 - 2023 CURVC Corp. All rights reserved

는 정보는 Version 명, 릴리즈 날짜, 설명, 상태, 공유, 위키 링크 등이며 해당 정보를 수정하거나 삭제할 수 있다. 10.5 Issue categories 이슈 카테고리는 이슈에 대한 카테고리를 지정하고 기본 Assignee를 할당할 수 있다. 10.6 Wiki Wiki 탭에서는 Wiki의 시작 페이지를 설정할 수 있다. 10.7 Repositories Repositories 탭에서는 프로젝트의 소스코드 저장소를 추가할 수 있다. 10.8 Forums Forums 탭에서는 프로젝트에서 사용할 수 있는 간단한 Forums을 생성할 수 있다. 다음 그림은 Forums 탭에 대한 예제 그림을 보여준다. image2017-2-22_20-28-29.png 10.9 Activities Activities 탭에서는 Time Tracking 기록을 위한 활동을 선택할 수 있다.

2017-03-01 00:55:48.69

Redmine 관리자 가이드

이 문서는 Redmine의 관리자에 대한 일반적인 가이드를 제공한다. 1. Projects Administration > Projects에서는 기본적으로 모든 Active 프로젝트의 리스트를 보여준다. 모든 프로젝트를 확인하기 위해서는 Filters의 Status를 All로 변경할 수 있다. 프로젝트 리스트에서 프로젝트를 선택하면 각 프로젝트의 settings로 이동하며, 프로젝트에 대한 정보를 수정할 수 있다. image2017-2-22_21-33-10.png 1.1 프로젝트 상태 레드마인에서 프로젝트 상태는 다음과 같이 분류 될 수 있다. Active : 프로젝트가 정상적으로 운영되고 있는 상태 Closed : 프로젝트가 Read Only인 상태 Archived : 사용자에게 더이상 보이지 않는 상태 1.2 프로젝트 생성 새로운 프로젝트를 만들기 위해 프로젝트 메뉴의 상단의 New project 링크를 선택한다. 혹은 프로젝트 리스트에서 복사를 희망하는 프로젝트에서 Copy 링크를 선택하면 해당 프로젝트의 정보를 복사하여 프로젝트를 생성할 수 있다. Name : 프로젝트 이름을 설정한다. Description : 프로젝트에 대한 설명을 설정한다. Identifier : 프로젝트 식별자를 설정한다. Homepage : 관련 홈페이지의 링크를 설정한다. Public : 프로젝트를 공용으로 설정한다. Subproject of : 서브프로젝트로 설정한다. Inherit members : 사용자를 상속 받는다. Modules : 프로젝트에 활성화하기를 원하는 모듈을 선택한다. Tracker : 프로젝트에서 사용하기 원하는 이슈 타입을 선택한다. Custom fields : 프로젝트에서 사용하기 원하는 커스텀 필드를 선택한다. image2017-2-22_21-35-4.png 1.3 프로젝트 삭제 프로젝트를 삭제를 위해 프로젝트 리스트에서 삭제를 원하는 프로젝트의 Delete 링크를 선택해준다. 2. Users Administration > Users에서 사용자 리스트와, 상태 관리, 정보수정, 추가, 삭제 등을 수행할 수 있다. 2.1 사용자 검색 기본적으로 active되어 있는 사용자가 표시되며, 전체 사용자 리스트를 보기위해서는 Filters에서 Status를 변경하면 된다. 다음 그림은 사용자 리스트 조회 화면을 보여준다. image2017-2-22_20-58-32.png 2.2 사용자 정보 수정 사용자 리스트에서 특정 사용자 Login 아이디를 선택하면, 해당 사용자의 정보를 수정할 수 있다. Information Login : 로그인 아이디 First name : 이름 Last name : 성 Email : 이메일 Language : 사용하고자 하는 언어 선택 Administrator : 관리자 권한 여부 Email notifications For any event on all my projects 나의 모든 프로젝트의 알림을 받음 For any event on the selected projects only... 선택된 프로젝트만 알림을 받음 Only for things I watch or I'm involved in 내가 포함되어 있거나 보고 있는 이슈 Only for things I am assigned to 내가 할당된 이슈 Only for things I am the owner of 내가 소유자인 이슈 No events 받지 않음 I dont want to be notified of changes that I make myself 내가 만든 수정에 대한 알림을 받지 않음 Preferences Hide my email address : 이메일 주소 숨김 Time zone : 표시를 원하는 시간 대역 Display comments : 코멘트 출력 방식 Warn me when leaving a page with unsaved text : 저장 알림 Authentication Authentication mode 내부 사용자 관리를 사용할지? 아니면 외부 Directory를 선택 2.3 사용자 추가 새로운 사용자를 추가하기 위해서는 사용자 리스트 페이지에서 New user 링크를 선택한다. 2.4 사용자 삭제 혹은 Lock 사용자를 삭제하기 위해서 사용자 리스트이 끝 컬럼에서 Delete 버튼을 선택한다. 사용자를 Lock하기 위해서는 사용자 리스트의 끝 부분 컬럼에서 Lock 버튼을 선택한다. 사용자 Lock은 해당 사용자의 로그인 및 접근 제한, 이슈 할당 제한, Watcher 제한, 이메일 노티 제한, 그리고 프로젝트 멤버에서 보여지지 않는다. 3. Groups Administration > Groups 에서는 레드마인에서 사용될 수 있는 그룹을 생성 및 삭제하고 해당 그룹에 사용자를 할당할 수 있다. 다음 그림은 Groups 페이지에 대한 예제를 보여준다. image2017-2-22_21-0-19.png 3.1 Group 생성 새로운 그룹을 생성하기 위해 Groups 리스트의 상단의 New group 링크를 선택한다. 적절한 그룹 명을 입력하고 하단 Create 버튼을 클릭한다. 생성한 그룹명은 리스트에서 변경을 원하는 특정 그룹을 선택하고 General 탭에서 그룹명 변경 후 Save 버튼을 클릭한다. 3.2 Group에 사용자 할당 그룹에 사용자를 할당하기 위해, 리스트에서 특정 그룹을 선택하고 Users 탭에서 New user 링크를 통해 사용자를 추가한다. 다음 그림은 Group에 사용자를 추가하는 예제를 보여준다. image2017-2-22_21-0-45.png 특정 그룹을 프로젝트로 할당하기 위해 Projects 탭에서 Add Projects 링크를 클릭 후 적용을 원하는 프로젝트와 Roles을 선택한다. 3.3 Group 삭제 특정 그룹을 삭제하기위해 프로젝트 리스트에 Delete 버튼을 클릭한다. 4. Roles and Permissions Administration > Roles and Permissions에서는 프로젝트에서 사용되는 사용자의 역할과 해당 역할에 대한 권한을 설정할 수 있다. image2017-2-22_21-1-17.png 4.1 새로운 역할 추가 새로운 역할을 추가하기 위해 상단 New role 링크를 선택한다. 공통 Name : 새로운 역할 이름 Issues can be assigned to this role : 역할에 이슈 할당 Issue visibility : 볼 수 있는 이슈들 Time logs visibility : 볼 수 있는 타임로그 Users visibility : 볼 수 있는 사용자 Copy workflow from : 워크플러우 복사 대상 Permissions Project : 프로젝트 관련 권한 Forums : 포럼 관련 권한 Calendar : 달력 관련 권한 Documents : 문서 관련 권한 Files : 파일 관련 권한 Gantt : Gantt 관련 권한 Issue tracking : 이슈 트래킹 관련

2014 - 2023 CURVC Corp. All rights reserved

권한 News : 뉴스 관련 권한 Repository : 저장소 관련 권한 Time tracking : 타임 트래킹 관련 권한 Wiki : 위키 관련 권한 image2017-2-22_21-1-38.png 4.3 Permissions Report Permissions Report를 통해 각 역할 별로 권한을 표로 보거나 부여할 수 있다. image2017-2-22_21-2-14.png 4.4 역할 삭제 각 역할은 역할 리스트에서 Delete 버튼을 통해 삭제할 수 있다. 5. Tracker Administration의 Tracker에서는 레드마인에서 사용하는 이슈 타입(Tracker)의 리스트를 확인할 수 있다. image2017-2-22_21-2-43.png 5.1 새로운 이슈타입 추가 새로운 이슈타입을 만들기 위해 상단 New tracker 링크를 선택한다. 공통 Name : 이슈타입의 이름 Default status : 이슈가 생성되었을 때 기본 상태 Issues displayed in roadmap : 로드맵에 표시 여부 Standar fields : 이슈에 사용할 표준 필드 Custom fields : 이슈에 사용할 커스텀 필드 Copy workflow from : 워크플로우를 복사할 대상 프로젝트 프로젝트에 영역에서는 해당 이슈를 적용할 프로젝트를 선택한다. 5.2 Summary 상단 Summary 링크를 선택하면, 다음 그림과 같이 이슈 타입 별로 사용하는 필드의 리스트를 보거나 선택할 수 있다. image2017-2-22_21-3-22.png 6. Issue statuses Administration의 Issue statuses에서는 Redmine 이슈에서 사용하는 이슈의 상태의 리스트를 확인할 수 있다. image2017-2-22_21-3-54.png 6.1 새로운 상태 추가 새로운 상태를 추가하기 위해서 상단 New status 링크를 선택한다. New status 화면이 나타나면 이슈 제목을 선택하고 해당 상태가 Closed를 나타내는지 여부를 체크해서 Create를 선택한다. 7. Workflow Administration의 Workflow에서는 역할 별 이슈 타입의 워크플로우를 선택할 수 있다. 7.1 Workflow 수정 워크플로우를 수정하기 위해 상단 Role에서 수정하고자하는 역할을 선택하고 Tracker에서 수정을 원하는 이슈를 선택한다. 좌측은 현재 상태이고 우측 상단의 상태들은 이동할 수 있는 상태를 나타낸다. 만약 진행에서 해결로 이동하기를 원하면 진행을 체크하면 된다. image2017-2-22_21-5-24.png 7.2 Fields Permissions Workflow 상태별로 접근할 수 있는 필드를 설정하기 위해서는 Fields permissions 탭을 선택한다. 이 역시 상태 별로 해당 필드를 Edit 모드로 할지? Read only로 할지?를 선택한다. image2017-2-22_21-5-48.png 8. Custom fields 레드마인에서는 기본 레드마인 표준 필드 외에 추가적인 정보를 포함할 수 있는 커스텀 필드를 제공한다. image2017-2-22_21-10-21.png 8.1 Custom fields 생성 새로운 커스텀 필드를 생성하기 위해 Administration 〉 Custom fields 〉 New custom field를 선택한다. New custom field 페이지가 나타나면 먼저 커스텀 필드의 범주를 선택한다. Select the type of object to which the custom field is to be attached : Issues : 이슈 Spent time : 시간 Projects : 프로젝트 Versions : 버전 Documents : 문서 Users : 사용자 Groups : 그룹 Activities : 활동 Issue priorities : 이슈 우선순위 Document categories : 문서 카테고리 image2017-2-22_21-10-35.png 다음 정보를 참고하여 필요한 정보를 선택한다. Format : 지원되는 Format은 다음과 같다. Boolean : 체크박스 Date : 날짜 Float : floating 포인트 수 Integer : 양수 혹은 음수 Link : URL List : 드랍다운 리스트 Long Text : 멀티라인 텍스트 User : 사용자 Version : 버전 Name : 필드명 Description : 설명 Min - Max length : 최소 - 최대 길이 Regular expression : 정규표현식 적용 Text formatting : 선택 시, 위키 문법 지원 Default value : 기본가격 Link values to URL : 다음과 같은 변수를 사용하는 URL 생성 %value% : 커스텀 필드 값 %id% : 커스텀 오브젝트의 아이디 %project_id% : 커스터마이즈된 오브젝트의 프로젝트 아이디 %project_identifier% : 커스터마이즈된 오브젝트의 프로젝트의 식별자 %m1%, %m2% : 커스텀 필드 정규식에 매치되는 그룹 캡처 Required : 이슈를 생성하거나 저장하기 위해 요구되는 필드 For all projects : 만약 체크된다면, 모든 프로젝트를 위해 사용된다. Searchable : 레드마인에서 검색 가능한지 여부 Visible : 사용자 프로파일 여부 image2017-2-22_21-7-40.png 9. Enumerations Enumeration은 Document categories, Issue priorities, Activities(time tracking)에서 사용되는 Enumerations 값을 설정한다. image2017-2-22_21-11-41.png 9.1 Enumerations 추가 각 Enumerations에 값을 추가하기 위해 하위에 New value 링크를 선택한다. 다음 항목을 참고하여 입력하고 Create 버튼을 클릭한다. Name : 새로운 Enumerations 값을 입력 Active : 체크 시 활성화 된다. Default value : 기본 값으로 설정한다. 9.2 Enumerations 삭제 리스트에서 삭제를 원하는 아이템에서 Delete 버튼을 클릭한다. 10. Settings Settings에서는 레드마인의 전역에 적용될 수 있는 다양한 설정을 변경할 수 있게 해준다. 설정을 위해 Administration 〉 Settings를 선택한다. 10.1 General General 탭에서는 다음에 대한 설정을 적용할 수 있다. Application title : 레드마인 헤딩에 나타나는 제목 Welcome text : 레드마인 홈페이지에 표시되는 텍스트로 HTML을 지원한다. Attachment max. size : 업로드 파일의 최대 사이즈 Objects per page options : Issues, Comments 등 페이지에 보여지는 기본 숫자 Days displayed on project activity : 프로젝트 엑티비티 내의 보여주는 일 수 Host name and path : 레드마인 서버의 호스트 이름과 경로 Protocol : 접속을 위한 프로토콜 http 혹은 https Text formatting : 텍스트 필드의 에디터를 위한 포멧 Cache formatted text : 포멧 텍스트의 캐싱 활성화 Wiki history compression : 위키 히스토리 저장소의 압축 활성화 Feed content limit : RSS 피드에 포함되는 최대 피드 수 Max size of text files displayed inline KB : 텍스트 파일의 최대 사이즈 Max number of diff lines displayed : 변경 라인 최대 수 image2017-2-22_21-13-8.png 10.2 Display Display 탭에는 다음에 대한 설정을 적용할 수 있다. Theme : 커스텀 테마를 선택할 수 있게 해준다. Default language : 기본 언어 설정 Date format : 날짜 포멧 Based on user's language : 사용자 언어에 따라 설정됨 Other formats : 사용자가 설정한 포멧 Time format : 타임 포멧 Based on user's language : 사용자 언어에 따라 설정됨 Other foramts : 사용자가 설정한 포멧 Users display format : username이 보여지는 방식 Use Gravatar user icons : Gravatars 사용자 아이콘 사용 Default Gravatar image : 기본 Garavatar 이미지 image2017-2-22_21-14-16.png 10.3 Authentication Authentication 탭에는 다음에 대한 설정을 적용할 수 있다. Authentication required : 체크되어 있다면, anonymous 사용자의 페이지 접근을 차단 Autologin : 자동으로 로그인 기간 혹은 기능 사용 여부 Self-registration : 이 옵션은 회원가입을 활성화 하거나 비활성화할 수 있음 Minimum password length : 최소 패스워드 길이 Lost password : 이 옵션이 체크되어 있다면, 패스워드 찾기 기능이 활성화 된다. Allow OpenID login and registration : OpenID 로그인과 등록 가능 여부 Session expiration : Session maximum lifetime : 세션의 최대 라이프 타임 설정 Session inactivity timeout : 비활동 시 세션 타임아웃 설정 image2017-2-22_21-14-30.png 10.4 Projects Project 탭에서는 다음에 대한 설정을 적용할 수 있

2014 - 2023 CURVC Corp. All rights reserved

다. New projects are public by default : 새로운 프로젝트의 공개 여부 Default enabled modules for new projects : 새로운 프로젝트를 위한 기본적으로 활성화 되는 모듈 설정 Default trackers for new projects : 새로운 프로젝트를 위한 기본 이슈 타입 설정 Generate sequential project identifiers : 이 옵션을 사용 시, 레드마인의 식별자를 순차적으로 생성함 Role given to a non-admin user who creates a project : 관리자가 아닌 역할에 프로젝트 생성 권한을 부여함. image2017-2-22_21-14-44.png 10.5 Issue tracking Issue tracking 탭에서는 다음에 대한 설정을 적용할 수 있다. Allow cross-project issue relations : 설정 시 다른 프로젝트로부터 이슈들 사이의 관계를 생성할 수 있다. Link issues on copy : 복사한 이슈의 링크 여부 Allow cross-project subtasks : 서브 태스크에 대한 제한 Allow issue assignment to groups : 이슈를 그룹에 할당하는 것을 허락 Use current date as start date for new issues : 새로운 이슈를 생성 시, 현재 시점을 시작 날짜로 사용 Display subprojects issues on main projects by default : 설정 시, 서브 프로젝트의 이슈는 메인 프로젝트의 리스트에 보여짐 Calculate the issue done ratio with : 이슈 완료 % 설정 Non-working days : 휴일 설정 Issues export limit : 이슈 Export 제한 Maximum number of items displayed on the gantt chart : 간트차트에 출력되는 최대 아이템의 수 Default columns displayed on the issue list : 이슈 리스트에 보여지는 기본 컬럼 image2017-2-22_21-15-0.png 10.6 Email notifications Email notifications 탭에서는 이메일 알림과 관련되 다음 설정을 적용할 수 있다. Emission email address : 알림 이메일의 From에 사용되는 이메일 주소 Blind carbon copy recipients(bcc) : 설정 시, 이메일은 숨은 참조로 알림을 보낸다. Plain text mail (no HTML) : 설정 시, 이메일은 text 모드로 보내진다. Default notification option : 기본 알림 설정 Select actions for which email notifications should be sent : 이메일 알림 액션 설정 Email header : 이메일 헤더 설정 Email footer : 이메일 푸터 설정 image2017-2-22_21-15-17.png 10.7 Incoming emails Incoming emails 탭에는 다음 설정을 적용할 수 있다. Truncate emails after one of these lines : incoming 메일로부터 signatures 제거 사용 Exclude attachments by name : 첨부파일 제거 Enable WS for incoming emails API key : 레드마인은 이메일로부터 이슈 생성 혹은 커멘트 작성을 구성할 수 있으며, 이를 위해 받는 메일의 API를 활성화해야 한다. image2017-2-22_21-16-15.png 10.8 Repositories Repositories 탭에는 다음 설정을 적용할 수 있다. Enabled SCM : 프로젝트에 연결할 수 있는 SCM 설정 Fetch commits automatically : 저장소로 부터 자동으로 새로운 버전의 리비전을 가져온다. Enable WS for repository management : 자동 SVN 저장소 생성 스크립트가 설치되어 있다면, 활성화할 수 있다. Maximum number of revisions displayed on file log : 파일 로그에 보여지는 최대 리비전 수 Referencing keywords : Commit 메시지에서 이슈와 연계를 위해 참조할 키워드 Allow issues of all the other projects to be referenced and fixed : 다른 프로젝트의 이슈와의 연계 Enable time logging : Commit 메시지로부터 타임 로깅 허락 ex) Implement feature #1234 @3h15m Activity for logged time : Time logging을 위한 Activity 설정 image2017-2-22_21-16-31.png 11. LDAP authentication 레드마인은 LDAP을 통한 사용자 인증 기능을 제공하고 있다. 이를 설정하기 위해 Administration > LDAP authentication 메뉴를 선택한다. image2017-2-22_21-16-55.png 11.1 New authentication mode 새로운 LDAP 인증 모드를 생성하기 위해 상단 New authentication mode를 선택한다. Name : 디렉토리 별칭 명 Host : LDAP 호스트 네임 Port : LDAP 포트 Account : LDAP에 읽기 접근을 위한 사용자이름 Password : 계정을 위한 패스워드 Base DN : LDAP 디렉토리 트리의 최상위 레벨의 DN LDAP filter : LDAP 적용을 위한 Filter Timeout : 타임아웃 시간 On-the-fly user creation : 체크 시, 특정 LDAP 사용자가 레드마인에 로그인 시 레드마인 계정이 생성됨 Login attribute : 로그인에 사용될 아이디 매핑 Firstname attribute : 이름 매핑 Lastname attribute : 성 매핑 Email attribute : 이메일 매핑 image2017-2-22_21-17-37.png 12. Plugins Plugins 메뉴에는 현재 설치되어 있는 플러그인의 리스트와 구성을 설정할 수 있게 해준다. image2017-2-22_21-18-16.png 13. Information 설치된 레드마인의 정보를 확인할 수 있다. image2017-2-22_21-18-42.png 14. 알림 메일 구성 이메일 구성은 configuration.yml에서 설정할 수 있다. 다음은 구글 이메일을 보내는 메일로 설정한 예제를 보여주고 있다. production: email_delivery: delivery_method: :smtp smtp_settings: enable_starttls_auto: true address: "smtp.gmail.com" port: '587' domain: "smtp.gmail.com" authentication: :plain user_name: "your_email@gmail.com" password: "your_password" 다음은 지원되는 이메일 인증(authentication) 방식을 보여준다. nil plain login cram_md5 다음은 지원되는 이메일 발송 방식(delivery_method)을 보여준다. smtp sndmail async_smtp async_sendmail 15. 참고도서 다음은 참고할 수 있는 레드마인 도서를 보여준다.

2017-02-20 22:11:32.8

4. Git On The Server

2014 - 2023 CURVC Corp. All rights reserved

2019-06-15 16:13:02.25

2. Git 협업하기

2017-08-17 14:31:39.849

Migration

2020-05-29 22:05:37.8

Git Tips

Bare 저장소를 일반 저장소로 변환하기 1. bare 저장소에 .git 폴더 생성 mkdir .git 2. 모든 파일/폴더를 .git 폴더로 이동 mv branches/ .git/ mv config .git/ mv FETCH_HEAD .git/ mv HEAD .git/ mv hooks/ .git/ mv logs/ .git/ mv objects/ .git/ mv packed-refs .git/ mv refs/ .git/ 3. core.bare 설정 git config --local --bool core.bare false 4. master 브랜치 checkout git reset --hard mkdir .git mv branches/ .git/ mv config .git/ mv FETCH_HEAD .git/ mv HEAD .git/ mv hooks/ .git/ mv logs/ .git/ mv objects/ .git/ mv packed-refs .git/ mv refs/ .git/ git config --local --bool core.bare false git reset --hard Git username, password 저장하기 영구 저장 방법 ~/.git-credentials 파일에 plain text로 저장됨 (파일에 대해 access 권한 관리 필요) $ git config --global credential.helper store 임시 저장 방법 캐시에 임시 저장됨 $ git config credential.helper cache <timeout> Commit 개수 얻기 To get a commit count for a revision (HEAD, master, a commit hash): $ git rev-list --count <revision> 예) $ git rev-list --count branch-1 $ git rev-list --count 3930540717b How to authenticate user via git pre-receive hook https://stackoverflow.com /questions/38015205/how-to-authenticate-user-via-git-pre-receive-hook read old new ref author=$(git log -1 $ref --pretty=%an) committer=$(git log -1 $ref --pretty=%cn) echo author:$author echo committer:$committer fifty_first_commits = list(repo.iter_commits('master', max_count=50)) assert len(fifty_first_commits) == 50 # this will return commits 21-30 from the commit list as traversed backwards master ten_commits_past_twenty = list(repo.iter_commits('master', max_count=10, skip=20)) assert len (ten_commits_past_twenty) == 10 assert fifty_first_commits[20:30] == ten_commits_past_twenty

2017-02-20 22:10:25.149

3. Git Branching

2019-06-15 14:14:09.475

1. Git 시작하기

2014 - 2023 CURVC Corp. All rights reserved

2018-07-30 15:29:55.298

Git 교육자료

2019-06-16 12:18:51.827

Pull Request 만들기

Git 가이드

2020-04-22 08:13:04.096

브랜치 사용하기

Git 가이드

2019-06-16 12:17:05.256

동기화하기

Git 가이드

2020-04-22 09:49:47.836

병합 전략 (Merge Strategies)

Git 가이드

2020-04-22 09:49:35.562

병합 충돌 해결하기 (Merge Conflicts)

2014 - 2023 CURVC Corp. All rights reserved

충돌 해소 절차는 다음과 같다. Step 1) 파일 확인 $ git status On branch master You have unmerged paths. (fix conflicts and run "git commit") (use "git merge --abort" to abort the merge) Unmerged paths: (use "git add <file>..." to mark resolution) both modified: merge.txt (충돌 파일) Step 2) 파일 수정 편집기를 통해 파일을 수정한다. $ cat merge.txt <<<<<<< HEAD this is some content to mess with content to append ======= totally different content to merge later >>>>>>> new_branch_to_merge_later Step 3) Add to index & commit git add <conflicted files> git commit -m <commit message> Step 4) Push to origin git push origin Git 가이드

2020-04-22 09:40:37.978

Git Merge

머지는 분기된 이력을 병합하는 방법으로 하나의 브랜치 변경 이력을 다른 브랜치에 병합한다. 다음은 병합의 일반적인 절차를 설명한다. Step 1) 병합될 브랜치로 전환 본 예는 Feature → mastre 병합을 고려한다. git checkout master Step 2) origin 동기화 git pull origin Step 3) 머지 머지 명령를 이용해 Feature 브랜치를 master 브랜치로 병합한다. --no-ff 옵션은 머지 커밋을 생성하고 싶은 경우 지정한다. git merge [--no-ff] Feature Screen Shot 2020-04-22 at 9.08.22 AM.png Setp 4) Feature branch 삭제 git branch -d Feature Fast-Forward Merge 아래 그림과 같이 머지 대상 (master)에 변경 이력이 없는 상태에서 Feature의 변경 이력을 병합하는 경우를 의미한다. Screen Shot 2020-04-22 at 9.23.43 AM.png 3-way Merge 머지 대상 (본 예의 경우 master)에 변경 이력이 존재하는 상태에서 Feature의 변경 이력을 병합하는 경우를 의미한다. Screen Shot 2020-04-22 at 9.22.55 AM.png Merge Conflict 3-way merge의 경우 동일한 파일에 같은 지점이 변경된 경우 병합 대상 (본 예의 경우 Feature 브랜치)와 master를 병합할 때 충돌이 발생할 수 있다. 이 때 git은 고유한 방법으로 해당 파일 위치에 출동을 표시한다. $ cat merge.txt <<<<<<< HEAD this is some content to mess with (현재 브랜치 변경 내용) content to append ======= totally different content to merge later >>>>>>> new_branch_to_merge_later 일반적인 해소 절차는 관련 페이지를 참조한다. Git 가이드

2020-04-22 08:33:06.023

Git Branch

Git 은 강력한 브랜치 기능을 제공하므로 브랜치 기능을 이해하는 것이 중요하다. 브랜치 생성 브랜치는 원격 또는 로컬 저장소에서 생성할 수 있고 저장소간 동기 가능하다. git 은 현재 브랜치의 HEAD를 기반으로 브랜치가 생성된다. git branch <branch> 브랜치 목록 보기 로컬 브랜치 목록 로컬 브랜치 목록을 표시한다. git branch 전체 브랜치 목록 로컬 및 원격 브랜치 목록을 표시한다. git branch -a 브랜치 삭제 git은 안전한 삭제와 강제 삭제 두 가지 삭제 옵션을 제공한다. 안전 옵션 머지 되지 않은 변경사항이 있는 경우 삭제 되지 않는다. git branch -d <branch> 강제 옵션 이 옵션은 머지 상태와 관계 없이 브랜치 삭제가 가능하다. git branch -D <branch> 브랜치 이름 변경 git은 브랜치 이름 변경을 지원한다. git branch -m <branch> Screen Shot 2020-04-22 at 8.13.33 AM.png Git 가이드

2020-04-22 09:07:22.629

Git Checkout

2014 - 2023 CURVC Corp. All rights reserved

git checkout은 파일, 커밋 그리고 브랜치를 전환하는 방법을 제공한다. 새로운 브랜치로 전환 신규 브랜치를 생성하여 해당 브랜치로 전환하는 방법을 제공한다. existing-branch가 주어진 경우 주어진 브랜치 기반으로 브랜치를 생성하며 생략된 경우 현재 브랜치 기반으로 생성한다. git checkout -b <new-branch> [existing-branch] 존재하는 브랜치 전환 기존에 존재하는 브랜치로 전환하는 방법을 제공한다. git checkout <branch> 파일 전환 브랜치의 특정 파일 또는 폴더 전환도 지원한다. git checkout master -- <path 또는 파일 경로> Detached HEAD Git은 HEAD을 이용해 현재 snapshot를 관리한다. 내부적으로 checkout은 HEAD이 가리키는 commit 또는 branch 을 변경한다. HEAD가 아닌 commit을 checkout하면 git은 detached HEAD 상태로 전환한다. detached HEAD 상태는 참조하는 브랜치가 없기 때문에 과거 커밋 내용을 확인하는 등의 용도에서 사용고 정상적인 기능 개발을 위해 사용하지 않도록 주의한다. Screen Shot 2020-04-22 at 8.50.30 AM.png Git 가이드

2020-04-22 07:12:26.777

Git Fetch

git fetch는 원격 저장소 변경사항을 로걸 저장소에 동기화 한다. -all 옵션은 모든 원격 저장소 정보 동기화 한다. git fetch --all <remote>를 지정하면 특정 remote 의 변경 내용을 동기화 하며, branch가 지정되면 특정 브랜치 정보만 동기화 한다. git fetch <remote> [branch] --dry-run 옵션은 로컬 저장소에 반영하지 않는다. git fetch <remote> [branch] Screen Shot 2020-04-22 at 6.50.43 AM.png Git 가이드

2020-04-22 08:00:41.018

Git Push

일반 기능 로컬 저장소의 커밋 내용은 push 동작을 통해 원격 저장소에 전송한다. branch가 지정되지 않으면 현재 로컬 저장소와 동일한 원격 브랜치에 정보를 업로드 (동기화) 한다. git push <remote> [branch] Screen Shot 2020-04-22 at 7.27.55 AM.png 강제 업로드 Git은 기본적으로 fast-forward 만 허용한다. --force 옵션을 사용해 non-fast-forward 로 push 할 수 있다. 이력에 대해 정확한 이해가 있을 때만 --force 옵션을 사용한다. git push <remote> --force 모든 로컬 브랜치 업로드 git push <remote> --all 태그 정보 업로드 git push 기본 동작은 태그 정보를 업로드 하지 않는다. git push <remote> --tags fast-forward vs. non-fast-forward git push는 원격 저장소의 보호를 위래 기본적 동작은 fast-forward 모드를 권장한다. fast-forward: 로컬 저장소가 원격 저장소와 동기가 이루어진 상태에서 업로드하는 모드 non-fast-forward: 로컬 저장소가 원격 저장소에 뒤쳐진 상태에서 업로드하는 모드 fast-forward.png Git 가이드

2020-04-22 07:26:30.879

Git Pull

git pull은 원격 저장소 정보를 로컬 working folder에 반영한다. 상태1) 로컬 저장소의 master 브랜치가 원격 저장소에 뒤쳐진 상태 Screen Shot 2020-04-22 at 7.16.31 AM.png pull 후) git pull origin master 원격 저장소의 A-B-C 커밋이 단일 머지 커밋 (H)로 저장된다. Screen Shot 2020-04-22 at 7.19.22 AM.png rebase 옵션을 이용한 pull) rebase 옵션이 주어진 경우 원격 저장소의 A-B-C 커밋이 로컬 저장소의 브랜치에 rebase 된다. git pull --rebase origin Screen Shot 2020-04-22 at 7.22.40 AM.png Git 가이드

2014 - 2023 CURVC Corp. All rights reserved

2020-04-22 07:00:40.959

Git Remote

origin remote는 로컬 저장소의 원본 저장소를 지칭하며 origin은 clone 절차에서 자동으로 설정된다. 리모트 저장소 목록 보기 $ git remote [-v] origin upstream other_users_repo 원격 저장소 추가 아래 명령으로 원격 저장소를 추가한다. $ git remote add fake_test https://bitbucket.com/upstream_user/reponame.git; [remote "remote_test"] url = https://bitbucket.com/upstream_user /reponame.git fetch = +refs/heads/*:refs/remotes/remote_test/* 원격 저장소 점검 git remote show upstream * remote upstream Fetch URL: https://bitbucket.com/upstream_user/reponame.git Push URL: https://bitbucket.com/upstream_user/reponame.git HEAD branch: master Remote branches: master tracked simd-deprecated tracked tutorial tracked Local ref configured for 'git push': master pushes to master (fast-forwardable) 원격 저장소와 로컬 저장소 동기 방법 Git은 원격 저장소와 로컬 저장소 동기 방법을 제공한다. 세부 사용법은 해당 설명 페이지를 참고한다. git fetch git pull git push Screen Shot 2020-04-22 at 6.50.43 AM.png Git 가이드

2017-08-17 14:51:49.397

Migrating A CVS Repository To GIT

Importing from a CVS repository can be done using git-cvsimport either by direct access to the remote CVS server or from a snapshot tarball. The procedure includes the following steps: Import a CVS repository to a git repository (ususally on a local machine) Create a bare clone of the latter to be used as a shared read/write repository (again locally) Set a new repository on the server through gitosis (see above "Starting a new project repository) Push the cloned one in 2. to the newly created one in 3. All steps above should be done on a local machine. 1.CVS repository import cvsimport 설치 필요 user@local# sudo yum install git-cvsimport Direct import from a server user@local# cvs -z3 -d :pserver:anoncvs@cvs.curvc.com:/home /gmx/cvs login user@local# git cvsimport -A ./authors.txt -v -a -i -k -d : pserver:anoncvs@cvs.curvc.com:/home/gmx/cvs -C gromacs-cvs.git cmx This way is very slow, and it is much faster to use a tarball of a CVS snapshot. authors.txt: CVS 사용자를 Git 사용자로 맵핑 형식 예) username1_in_cvs = Git User1 <gituser1@curvc.com> username2_in_cvs = Git User2 <gituser2@curvc.com http://curvc.com> (Alternative) Import from a CVS snapshot tarball rossen@cvs.curvc.com# cd /home/gmx rossen@cvs. curvc.com# tar cvfz /home/rossen/gromacs-cvs-20090218.tar.gz cvs rossen@local# pwd /home/rossen/temp rossen@local# scp cvs.curvc.com: gromacs-cvs-20090218.tar.gz . rossen@local# tar xvfz gromacs-cvs-20090218.tar.gz rossen@local# git-cvsimport -a -i -k -d /home/rossen/temp /cvs -C gromacs-cvs.git gmx 2. Create a bare clone of the imported GIT rossen@local# mkdir gromacs rossen@local# cd gromacs rossen@local# git --bare init --shared rossen@local# git --bare fetch ../gromacs-cvs.git master: master rossen@local# git remote add gromacs git@git.curvc.com:gromacs. git 3. Set a new repository on the server Git 서버에서 gromacs repository 생성 4. Push the clone to the server rossen@local# git push gromacs --mirror Links git-cvsmigration http://www.kernel.org/pub/software/scm/git/docs /gitcvs-migration.html git-cvsserver http://www.kernel.org/pub/software /scm/git/docs/git-cvsserver.html git-cvsimport http://www.kernel.org/pub /software/scm/git/docs/git-cvsimport.html

2021-07-07 17:09:11.724

Git Push Store In Secure Store가 동작하지 않을 때

2014 - 2023 CURVC Corp. All rights reserved

현상 Git push 동작 시 계정정보를 계속 해서 물어보는 현상 (store in secure store 를 체크했을때도) 원인 어떠한 원인으로 인하여 git 저장소의 key가 꼬여버림 해결방안 Windows > Preferences > General > Security > Secure Storage > Contents 에서 git 저장소 선택 후 Delete image2021-7-7_17-5-20.png Eclipse 재시작 다시 commit&push하면서 계정정보 입력 및 store in secure store 체크하여 push하면 정상적으로 저장됨

2023-08-03 13:17:54.674

Git 개행처리 설정 방법

이 문서는 Git 사용중 OS 간의 개행 처리 방법이 다른 문제를 해결하는 개행 처리 설정 방법에 대한 가이드를 공유하기 위해 작성되었다. 개요 Git config 적용 범위는 시스템 설정, 글로벌 설정, git 저장소별 설정이 가능하고 우선순위는 git 저장소 → 글로벌 → 시스템 순이다. (git 저장소가 가장 높음) 특별하게 저장소에 별도로 설정할 일이 있는 경우를 제외하고 일반적으로 사용자 레벨인 글로벌에 config를 설정한다. Config 파일 위치 : git저장소 : 저장소₩.git₩config 글로벌 : C:₩Users₩사용자계정₩.gitconfig 시스템 : C:₩ProgramData₩Git₩config autocrlf 설정 사용자 각자가 git bash에서 아래 config 설정 입력. .gitattributes의 경우 eclipse의 egit에서 해당 파일을 읽을 수 없어서 미사용. Windows 환경 윈도우에서는 CRLF 를 사용하므로 저장소에서 가져올 때 LF 를 CRLF 로 변경하고 저장소로 보낼 때는 CRLF 를 LF로 변경하도록 true로 설정한다. git config --global core.autocrlf true Linux, MAC, Unix 환경 리눅스, 맥, 유닉스는 LF 만 사용하므로 input 으로 설정한다. git config --global core.autocrlf input 파일모드 false 설정 git이 리눅스의 파일모드를 감시하기 때문에 파일 권한으로 인하여 파일이 git status에 변경된 이력으로 노출될 수 있음. 다음 설정을 통해서 filemode를 false 설정한다. git config --global core.filemode false git config 설정 적용 확인 git config --list Eclipse에서 Git config 적용 일반적으로 Eclipse의 git config는 git의 config를 바라보고 있으나 Eclipse에서도 git config 설정이 가능하다. 경로: 메뉴 > Window > Perference > Team > Git > Configuration User Settings 탭에서 Add Entry 등록 key : autocrlf / Value : true Key : filemode / Value : false crlf 변경하여 다시 적용하는 방법 git repository를 새로 Clone 받았을때 아무런 변경도 하지 않았는데 Changes에 파일들이 변경이 되었다고 나오면 개행이 달라서 변경이 된 것으로 노출되는 것이다. git에서 관리하는 파일의 개행이 깨졌을 경우 복구하는 방법은 다음과 같다. 본인의 작업 환경 crlf 설정 확인(Window의 경우 crlfauto = true) eclipse에서 clone import(git bash나 다른 클라이언트에서 clone진행시 클라이언트의 기능에 의해서 자동으로 개행처리해줘서 문제가 발생하지 않을 수 있음) workspace에서 git bash로 접속 git status 실행 시 개행으로 인하여 change로 변경된 파일들 리스트가 출력 git add . (.project파일을 제거) git commit -m "crlf change" git push commit change에서 + -가 개수 동일한지 확인 CRLF / LF 보기상 차이점 git diff를 입력하면 파일의 개행 변경 내용을 확인 할 수 있다. [CRLF -> LF로 변경된 예시] -This is line 1^M -This is line 2^M -This is line 3^M +This is line 1 +This is line 2 +This is line 3 [LF -> CRLF로 변경된 예시] -This is line 1 -This is line 2 -This is line 3 +This is line 1^M +This is line 2^M +This is line 3^M

2022-04-25 21:33:52.426

Git Diff 두 점(..)과 세점 (...) 옵션 차이

2014 - 2023 CURVC Corp. All rights reserved

이 페이지는 Git의 diff 명령의 옵션 중 두 점 (..) 과 세 점 (...)의 차이점에 대해 정리한다. A---B---C topic / D---E---F---G master 커밋 A: a.txt 파일 수정 커밋 B: b.txt 파일 추가 커밋 C: c.txt 파일 삭제 커밋 F: f.txt 파일 변경 커밋 G: g.txt 파일 추가 위와 같이 커밋 이력이 발생한 경우, 두 점 (..) 또는 점 없는 옵션 master → topic git diff master topic --name-status 또는 git diff master..topic --name-status 은 F, G 커밋을 포함한 master 브랜치와 A, B, C 커밋을 포함한 topic 브랜치 사이의 차이점을 비교한다. 즉, 결과는 master (커밋 G)를 기준으로 topic 관점에서 차이점을 나타낸다. M a.txt A b.txt D c.txt M f.txt D g.txt topic → master git diff topic master --name-status 또는 git diff topic..master --name-status 은 F, G 커밋을 포함한 master 브랜치와 A, B, C 커밋을 포함한 topic 브랜치 사이의 차이점을 비교한다. 즉, 결과는 topic (커밋 C)를 기준으로 master 브랜치 관점에서 차이점을 나타낸다. M a.txt D b.txt A c.txt M f.txt A g.txt 세 점 (...) 옵션 master → topic git diff master...topic --name-status 은 master 브랜치와 topic 브랜치의 가장 최근 공통 커밋을 기준으로 topic 브랜치 관점에서 차이점을 비교한다. 즉, 결과는 커밋 E를 기준으로 topic 브랜치 (커밋 C) 관점에서 차이점을 나타낸다. M a.txt A b.txt D c.txt topic → master git diff topic...master --name-status 은 topic 브랜치와 master 브랜치의 가장 최근 공통 커밋을 기준으로 master 브랜치 관점에서 차이점을 비교한다. 즉, 결과는 커밋 E를 기준으로 master 브랜치 (커밋 G) 관점에서 차이점을 나타낸다. M f.txt A g.txt

2021-07-05 12:44:12.31

특정 파일 이력 원복하기 (Eclipse)

파일 단위 배포에서 변경목록 작업 중 Commit 또는 Push 된 파일들을 원복 시키는 방법을 가이드 합니다. 원복 방법 방법1) 수동으로 소스 수정하여 다시 commit&push하면 변경목록에서 파일 제거 가능 SR 브랜치가 생성된 시점과 완전히 동일해야 함 Whitespace, EOL도 일치해야함 방법2) Eclipse 가 제공하는 기능 활용하여 파일 내용 원복 원복 대상 파일 선택 Replace 메뉴선택 원복 대상 파일에 대해 마우스 우클릭 > Replace With 메뉴 선택 원목 대상 종류 선택 Replace With 메뉴 선택 > Commit... 선택 작업 중인 파일 삭제 확인 창 Replace With 기능은 지정한 이력으로 로컬 파일을 원복하므로 선택한 파일의 백업이 필요하다면 취소하여 백업 후 절차를 다시 시작한다. 진행 확인 파일이 복원되어도 좋다면 "Discard Changes" 클릭 복원할 커밋 선택 변경 목록의 파일 제거 목적이므로 SR 브랜치가 생성된 부모 커밋 (또는 SR의 최초 커밋의 이전 커밋) 선택 확인 OK 버튼을 클릭하여 파일 복원 완료 복원한 파일을 Git Add, Commit, Push 하여 삭제 중인 변경목록 내의 파일 제거 image2021-7-5_12-44-5.png

2021-07-05 12:46:51.156

특정 파일 이력 원복하기 (Intellij)

파일 단위 배포에서 변경목록 작업 중 Push 된 파일들을 원복 시키는 방법을 가이드 합니다. 원복 방법 방법1) 수동으로 소스 수정하여 다시 commit&push하면 변경목록에서 파일 제거 가능 방법2)File history에서 get으로 원복 후 commit&push하면 변경목록에서 파일 제거 가능 image2021-7-1_17-12-29.png image2021-7-1_17-13-10.png

2021-07-07 12:38:48.093

Git 기본 환경 설정

2014 - 2023 CURVC Corp. All rights reserved

이 페이지는 개발 환경 중 Git의 기본 환경 설정하는 방법을 가이드한다. Git 환경 설정 파일 위치 Git 환경 파일은 다음과 같이 구별된다. --system: 시스템 default (${prefix}/etc/gitconfig), 모든 사용자에게 적용 --global: 사용자 default (~/.gitconfig), 사용자에게 적용 --local: 현재 repository default(.git/config), 현재 작업중인 reporitory에 적용 사용자 정보 설정 커밋을 기록하는 사용자 정보를 설정한다. Git 환경 설정 파일에 아래와 같은 형식으로 설정한다. [user] name = username email = username@khmnc.or.kr 라인 개행 문자 처리 설정 checkout 과 add 시에 라인 개행 문자 (CR 또는 CRLF) 처리 방법을 설정한다. 설정하지 않으면 시스템에서 사용하는 개행 문자가 적용되어 개발자 환경에 따라 파일을 구성하는 라인의 개행 문자가 섞일 수 있다 (Linux: LF, Windows: CRLF). LF 로 저장소의 개행 문자를 통일하는 것을 권장한다. 저장소의 개행 문자를 LF로 유지할 때의 aotocrlf 설정: input: 입력 대로 처리 (Linux, Mac, Cygwin 환경) true: 자동으로 LF로 변환 (Windows) [core] autocrlf = true

2020-11-26 07:49:54.47

(i) 이력에서 파일 삭제하기

2014 - 2023 CURVC Corp. All rights reserved

original site = https://rtyley.github.io/bfg-repo-cleaner/ https://rtyley.github.io/bfg-repo-cleaner/ an alternative to git-filter-branch The BFG is a simpler, faster alternative to git-filter-branch http://git-scm.com/docs/git-filter-branch for cleansing bad data out of your Git repository history: Removing Crazy Big Files Removing Passwords, Credentials & other Private data The git-filter-branch command is enormously powerful and can do things that the BFG can't - but the BFG is much better for the tasks above, because: Faster https://rtyley.github.io/bfg-repo-cleaner/#speed : 10 - 720x faster Simpler https://rtyley.github.io/bfg-repo-cleaner/#examples : The BFG isn't particularily clever, but is focused on making the above tasks easy Beautiful : If you need to, you can use the beautiful Scala language to customise the BFG. Which has got to be better than Bash scripting at least some of the time. Usage First clone a fresh copy of your repo, using the --mirror http://stackoverflow.com/q/3959924/438886 flag: $ git clone --mirror git://example.com/some-big-repo.git This is a bare http://git-scm.com/docs/gitglossary.html#def_bare_repository repo, which means your normal files won't be visible, but it is a full copy of the Git database of your repository, and at this point you should make a backup of it to ensure you don't lose anything. Now you can run the BFG to clean your repository up: $ java -jar bfg.jar https://repo1.maven.org/maven2/com/madgag/bfg/1.13.0/bfg-1.13.0.jar --strip-blobs-bigger-than 100M some-big-repo.git The BFG will update your commits and all branches and tags so they are clean, but it doesn't physically delete the unwanted stuff. Examine the repo to make sure your history has been updated, and then use the standard git gc http://git-scm.com/docs/git-gc command to strip out the unwanted dirty data, which Git will now recognise as surplus to requirements: $ cd some-big-repo.git $ git reflog expire --expire=now --all && git gc --prune=now --aggressive Finally, once you're happy with the updated state of your repo, push it back up (note that because your clone command used the --mirror flag, this push will update all refs on your remote server): $ git push At this point, you're ready for everyone to ditch their old copies of the repo and do fresh clones of the nice, new pristine data. It's best to delete all old clones, as they'll have dirty history that you don't want to risk pushing back into your newly cleaned repo. Examples In all these examples bfg is an alias for java -jar bfg.jar. Delete all files named 'id_rsa' or 'id_dsa' : $ bfg --delete-files id_{dsa,rsa} my-repo.git Remove all blobs bigger than 50 megabytes : $ bfg --strip-blobs-bigger-than 50M my-repo.git Replace all passwords listed in a file (prefix lines 'regex:' or 'glob:' if required) with ***REMOVED***wherever they occur in your repository : $ bfg --replace-text passwords.txt my-repo.git Remove all folders or files named '.git' - a reserved filename https://github.com/git/git/blob/d29e9c89d/fsck.c#L228-L229 in Git. These often become a problem http://stackoverflow.com/q/16821649/438886when migrating to Git from other source-control systems like Mercurial : $ bfg --delete-folders .git --delete-files .git --no-blob-protection my-repo.git For further command-line options, you can run the BFG without any arguments, which will output text like this https://repository.sonatype.org/service/local/artifact/maven/redirect?r=central-proxy&g=com.madgag&a=bfg&v=LATEST&e=txt. Your current files are sacred... By default the BFG doesn't modify the contents of your latest commit on your master (or 'HEAD') branch, even though it will clean all the commits before it. That's because your latest commit is likely to be the one that you deploy to production, and a simple deletion of a private credential or a big file is quite likely to result in broken code that no longer has the hard-coded data it expects - you need to fix that, the BFG can't do it for you. Once you've committed your changes- and your latest commit is clean with none of the undesired data in it - you can run the BFG to perform it's simple deletion operations over all your historical commits. Note: Cleaning Git repos is about completely eradicating bad stuff from history. If something 'bad' (like a 10MB file, when you're specifying --strip-blobs-bigger-than 5M) is in a protected commit, it won't be deleted - it'll persist in your repository, even if the BFG deletes if from earlier commits https://github.com/rtyley/bfg-repo-cleaner/issues/53#issuecomment-50088997. If you want the BFG to delete something you need to make sure your current commits are clean. Note that although the files in those protected commits won't be changed, when those commits follow on from earlier dirty commits, their commit ids will change, to reflect the changed history - only the SHA-1 id of the filesystem-tree will remain the same. If you want to turn off the protection (in general, not recommended) you can use the --no-blob-protection flag: $ bfg --strip-biggest-blobs 100 --no-blob-protection repo.git

2019-06-15 16:27:10.384

변경 저장하기

2014 - 2023 CURVC Corp. All rights reserved

'체크인'은 중앙 서버로 원격 푸시하는 작업입니다. 이것은 SVN 커밋이 프로젝트 변경 사항을 완전히 '저장'하기 위해 인터넷 액세스가 필요하다는 것을 의미합니다. Git 커밋은 로컬에서 캡처 및 빌드 한 다음 필요에 따라 git push -u origin master 명령을 사용하여 원격 서버에 푸시 할 수 있습니다. 두 가지 방법의 차이점은 아키텍처 설계 간의 근본적인 차이입니다. Git은 분산 애플리케이션 모델이지만 SVN은 중앙 집중식 모델입니다. 분산 응용 프로그램은 중앙 집중식 서버와 같이 단일 지점에서 장애가 발생하지 않으므로 일반적으로 더 강력합니다. git add, git status, git commit 명령은 모두 Git 프로젝트의 현재 상태의 스냅 샷을 저장하는 데 사용됩니다. Git은 'stash'라는 추가 저장 메커니즘을 가지고있다. stash는 커밋 할 준비가되지 않은 변경 사항을 임시로 저장하는 영역입니다. stash는 세 개의 트리 중 첫 번째 트리 인 작업 디렉토리에서 작동하며 광범위한 사용 옵션이 있습니다. 자세한 내용은 git stash 페이지를 참조하십시오. Git 저장소는 특정 파일이나 디렉토리를 무시하도록 설정할 수 있습니다. 이것은 자식이 무시한 내용의 변경 사항을 저장하지 못하도록합니다. 힘내는 무시 목록을 관리하는 여러 가지 설정 방법을 가지고있다. Git ignore configure에 대해서는 git ignore 페이지에서 자세히 설명합니다. Git 가이드

2019-06-15 23:11:01.843

저장소 점검하기

Git 가이드

2019-06-16 06:59:47.491

변경 취소하기

Git 가이드

2019-06-16 13:01:34.804

Rewriting History

Git 가이드

2020-04-22 05:47:33.397

Git 저장소

Git은 새로운 저장소 생성과 기존 저장소 복제를 제공합니다. image2020-4-22_5-47-3.png 저장소 란? Git 저장소는 프로젝트의 가상 저장소입니다. 필요한 경우 액세스 할 수있는 코드 버전을 저장할 수 있습니다. 새로운 저장소 생성하기: git init 새 저장소를 만들려면 git init 명령을 사용합니다. git init은 새로운 repo의 초기 설정 중에 사용하는 일회성 명령입니다. 이 명령을 실행하면 현재 작업 디렉토리에 새 .git 하위 디렉토리가 만들어집니다. 그러면 새로운 마스터 브랜치가 작성됩니다. Versioning an existing project with a new git repository 이 예에서는 기존에 프로젝트 폴더를 만들었으며 프로젝트 내에 repo를 만들려한다고 가정합니다. 먼저 루트 프로젝트 폴더로 이동 한 다음 git init 명령을 실행합니다. cd /path/to/your /existing/code git init git init을 기존 프로젝트 디렉토리로 지정하면 위에서 언급

2014 - 2023 CURVC Corp. All rights reserved

한 것과 동일한 초기화 설정이 실행되지만 해당 프로젝트 디렉토리로 범위가 지정됩니다. git init <project directory> 기존 저장소 복제하기: git clone 프로젝트가 중앙 저장소에 이미 설정된 경우 clone 명령은 사용자가 로컬 개발 복제본을 얻는 가장 일반적인 방법입니다. git init과 마찬가지로, 복제는 일반적으로 일회성 작업입니다. 개발자가 작업 복사본을 얻으면 모든 버전 제어 작업은 로컬 저장소를 통해 관리됩니다. git clone <repo url> git clone은 원격 리포지토리의 복사본이나 복제본을 만드는 데 사용됩니다. git clone에 저장소 URL을 전달합니다. Git은 몇 가지 다른 네트워크 프로토콜과 해당 URL 형식을 지원합니다. HTTP 프로토콜를 지원하는 Bitbucket 저장소 URL 형식 예: https://user@bitbucket.almdemo.curvc.com/scm/gt /spring3example.git https://user@bitbucket.almdemo.curvc.com/scm/gt /spring3example.git user: Bitbucket username bitbucket.almdemo.curvc. com/scm http://bitbucket.almdemo.curvc.com/scm: Bitbucket URL gt: Bitbucket project key spring3example: 저장소 이름 실행되면 마스터 브랜치의 원격 repo 파일의 최신 버전이 풀다운되어 새 폴더에 추가됩니다. 새 폴더의 이름은이 경우 javascript-data-store의 REPONAME 다음에 지정됩니다. 이 폴더는 원격 저장소와 새로 생성 된 master 브랜치의 전체 히스토리를 포함합니다. 저장소에 변경사항 저장하기: git add and git commit 저장소가 설정되면 이제 저장소를 복제하거나 초기화 했으므로 파일 버전 변경을 커밋 할 수 있습니다. 다음 예제에서는 / path / to / project에 프로젝트를 설정했다고 가정합니다. 이 예제에서 수행되는 단계는 다음과 같습니다. 디렉토리를 / path / to / project로 변경합니다. 내용이 담긴 CommitTest.txt 파일을 만듭니다 ~ "git tutorial for test content"~ git은 CommitTest.txt를 저장소 스테이징 영역에 추가합니다. 커밋에서 수행 된 작업을 설명하는 메시지로 "CommitTest.txt 파일 추가" 지정 cd /path/to/project echo "test content for git tutorial" >> CommitTest.txt git add CommitTest.txt git commit -m "CommitTest.txt 파일 추가" 이 예를 실행하면 Repo에 기록에 CommitTest.txt가 추가되고 파일의 향후 업데이트를 추적합니다. 이 예제는 두 개의 추가적인 git 명령 인 add와 commit를 소개했다. 이것은 매우 제한된 예 였지만, 두 명령 모두 git add 및 git commit 페이지에서 자세히 다루어집니다. git add의 또 다른 일반적인 사용 예는 --all 옵션입니다. git add --all을 실행하면 repo에서 변경되거나 추적되지 않은 파일을 가져 와서 repo에 추가하고 repo의 작업 트리를 업데이트합니다. 저장소 간 협업: git push Git의 "작업 카피"에 대한 아이디어가 SVN 저장소에서 소스 코드를 체크 아웃하여 얻은 작업 카피와 매우 다르다는 것을 이해하는 것이 중요합니다. SVN과 달리 Git은 작업 사본과 중앙 저장소를 구분하지 않습니다. 그들은 모두 본격적인 Git 저장소입니다. 이것은 Git과 SVN과 근본적으로 다른 협력을 만든다. SVN이 중앙 저장소와 작업 사본 사이의 관계에 의존하는 반면, Git의 공동 작업 모델은 저장소와 저장소 간의 상호 작용을 기반으로합니다. 작업 복사본을 SVN의 중앙 저장소로 검사하는 대신 한 저장소에서 다른 저장소로 커밋을 푸시하거나 끌어 당깁니다. 물론, Git repos라는 특별한 의미를주는 것을 막을 수있는 방법은 없습니다. 예를 들어 Git repo를 "중앙"저장소로 지정하면 Git을 사용하여 중앙 집중식 워크 플로를 복제 할 수 있습니다. 이는 VCS 자체에 내장되어 있지 않고 규칙을 통해 수행됩니다. Bare vs. cloned repositories 이전의 "새 저장소 초기화"절에서 git clone을 사용하여 로컬 저장소를 설정 한 경우 저장소는 원격 협업을 위해 이미 구성되어 있습니다. git clone은 복제 한 Git URL을 가리키는 리모컨으로 자동으로 저장소를 구성합니다. 즉, 파일을 변경하고 커밋하면 원격 저장소에 변경 사항을 적용 할 수 있습니다. git init을 사용하여 새로운 repo를 작성했다면 변경 사항을 푸시 할 원격 저장소가 없습니다. 새로운 repo를 초기화 할 때 일반적인 패턴은 Bitbucket과 같은 호스팅 된 Git 서비스로 가서 repo를 생성하는 것입니다. 이 서비스는 Git URL을 제공하여 로컬 Git 저장소에 추가하고 호스트 된 저장소로 Push 할 수 있습니다. 선택한 서비스로 원격 저장소를 만든 후에는 매핑을 사용하여 로컬 저장소를 업데이트해야합니다. 이 프로세스는 아래의 구성 및 설정 안내서에서 설명합니다. 자신의 원격 저장소를 호스팅하려면 "기본 저장소"를 설정해야합니다. git init과 git clone 모두 --bare 인수를 허용합니다. 베어 레포의 가장 일반적인 사용 사례는 원격 중앙 저장소를 만드는 것입니다. 설정: git config 원격 저장소 설정이 끝나면 로컬 저장소에 원격 저장소 URL을 추가하고 로컬 분기에 대한 업스트림 브랜치를 설정해야합니다. git remote 명령은 그러한 유틸리티를 제공합니다. git remote add <remote_name> <remote_repo_url> 이 명령은 <remote_repo_url>의 원격 저장소를 <remote_name>의 로컬 저장소에있는 참조로 매핑합니다. 원격 저장소를 매핑하면 로컬 저장소를 해당 저장소로 푸시 할 수 있습니다. git push -u <remote_name> <local_branch_name> 이 명령은 <local_branc_name> 아래의 로컬 repo 분기를 <remote_name>의 원격 저장소로 푸시합니다. git remote에 대한 자세한 내용은 Git 원격 페이지를 참조하십시오. 원격 repo URL을 설정하는 것 외에도 username 또는 email과 같은 글로벌 Git 구성 옵션을 설정해야 할 수도 있습니다. git config 명령을 사용하여 명령 행에서 Git 설치 (또는 개별 저장소)를 구성 할 수 있습니다. 이 명령은 사용자 정보에서 환경 설정, 저장소 작동에 이르기까지 모든 것을 정의 할 수 있습니다. 몇 가지 일반적인 구성 옵션은 다음과 같습니다. Git은 개별 저장소 (로컬), 사용자 (전역) 또는 전체 시스템 (시스템)에 범위 옵션을 제공하는 구성 옵션을 3 개의 개별 파일에 저장합니다. 로컬 : <repo> /. git / config - 저장소별 설정. 전역 : /.gitconfig - 사용자 별 설정. 여기에는 --global 플래그로 설정된 옵션이 저장됩니다. 시스템 : $ (접두사) / etc / gitconfig - 시스템 전체 설정. 현재 저장소의 모든 커밋에 사용할 작성자 이름을 정의하십시오. 일반적으로 --global 플래그를 사용하여 현재 사용자에 대한 구성 옵션을 설정하려고합니다. git config --global user.name <name> 현재 사용자가 모든 커밋에 사용할 작성자 이름을 정의하십시오. --local 옵션을 추가하거나 설정 수준 옵션을 전달하지 않으면 현재 로컬 저장소에 대한 user.name http://user.name이 설정됩니다. git config --local user. email <email> 현재 사용자가 모든 커밋에 사용할 작성자 전자 메일을 정의합니다. git config --global alias.<alias-name> <git-command> Git 명령에 대한 바로 가기를 만듭니다. 이것은 일반적으로 사용되는 git 명령에 대한 사용자 정의 바로 가기를 작성하는 강력한 유틸리티입니다. 단순한 예는 다음과 같습니다. git config --global alias.ci commit 이렇게하면 git commit 바로 가기로 실행할 수있는 ci 명령이 만들어집니다. git 별칭에 대한 자세한 내용은 git config 페이지를 참조하십시오. git config --system core.editor <editor> 현재 컴퓨터의 모든 사용자에 대해 git commit과 같은 명령이 사용하는 텍스트 편집기를 정의하십시오. <editor> 인수는 원하는 편집기 (예 : vi)를 시작하는 명령이어야합니다. 이 예제는 --system 옵션을 소개합니다. --system 옵션은 전체 시스템의 구성을 설정합니다. 즉, 모든 사용자와 시

2014 - 2023 CURVC Corp. All rights reserved

스템의 repos를 의미합니다. 구성 수준에 대한 자세한 내용은 git config 페이지를 참조하십시오. git config --global --edit 수동 편집을 위해 텍스트 편집기에서 글로벌 구성 파일을 엽니 다. git이 사용할 텍스트 편집기를 구성하는 방법에 대한 자세한 안내는 Git config 페이지에서 찾을 수 있습니다. Git은 구성 옵션을 세 개의 개별 파일에 저장하므로 개별 저장소, 사용자 또는 전체 시스템에 대한 옵션을 범위 지정할 수 있습니다. <repo> /. git / config - 저장소 별 설정. ~ / .gitconfig - 사용자 별 설정. 여기에는 --global 플래그로 설정된 옵션이 저장됩니다. $ (접두사) / etc / gitconfig - 시스템 전체 설정 이 파일의 옵션이 충돌하면 로컬 설정이 시스템 설정보다 우선하는 사용자 설정보다 우선합니다. 이러한 파일 중 하나를 열면 다음과 같은 내용이 표시됩니다. [user] name = John Smith email = john@example.com [alias] st = status co = checkout br = branch up = rebase ci = commit [core] editor = vim 이 값을 수동으로 git config와 동일한 효과로 편집 할 수 있습니다. Git 가이드

2019-06-15 17:12:02.931

Git Diff

변경 비교하기: git diff Diffing은 두 개의 입력 데이터 집합을 가져 와서 변경 내용을 출력하는 함수입니다. git diff는 실행시 Git 데이터 소스에서 diff 함수를 실행하는 다중 사용 Git 명령입니다. 이러한 데이터 소스는 커밋, 분기, 파일 등이 될 수 있습니다. 이 문서는 git diff와 diffing 작업 흐름 패턴의 일반적인 호출에 대해 설명합니다. git diff 명령은 git status 및 git log와 함께 사용되어 Git repo의 현재 상태를 분석합니다. Reading diffs: outputs Raw output format 다음 예제는 간단한 repo에서 실행됩니다. repo는 아래 명령으로 생성됩니다. $ mkdir diff_test_repo $ cd diff_test_repo $ touch diff_test.txt $ echo "this is a git diff test example" > diff_test.txt $ git init . Initialized empty Git repository in /Users/kev/code /test/.git/ $ git add diff_test.txt $ git commit -am"add diff test file" [master (root-commit) 6f77fc3] add diff test file 1 file changed, 1 insertion(+) create mode 100644 diff_test.txt 이 시점에서 git diff를 실행하면 결과가 출력되지 않습니다. diff에 대한 변경 사항이 없으므로 이는 예상되는 동작입니다. repo가 생성되고 diff_test.txt 파일이 추가되면 diff 출력을 실험하기 위해 파일 내용을 변경할 수 있습니다. $ echo "this is a diff example" > diff_test.txt 이 명령을 실행하면 diff_test.txt 파일의 내용이 변경됩니다. 일단 수정되면 diff를보고 출력을 분석 할 수 있습니다. 이제 git diff를 실행하면 다음과 같은 결과가 출력됩니다. $ diff --git a /diff_test.txt b/diff_test.txt index 6b0c6cf..b37e70a 100644 --- a/diff_test. txt +++ b/diff_test.txt @@ -1 +1 @@ -this is a git diff test example +this is a diff example 이제 diff 출력에 대한 자세한 분석을 살펴 보겠습니다. 1. Comparison input $ diff --git a/diff_test.txt b/diff_test.txt 이 행은 diff의 입력 소스를 표시합니다. a / diff_test.txt와 b / diff_test.txt가 diff에 전달되었음을 알 수 있습니다. 2. Meta data index 6b0c6cf..b37e70a 100644 이 행은 내부 Git 메타 데이터를 표시합니다. 이 정보가 필요 없을 것입니다. 이 출력의 숫자는 Git 객체 버전 해시 식별자에 해당합니다. 3. Markers for changes --- a/diff_test.txt +++ b /diff_test.txt 이 행은 각 diff 입력 소스에 기호를 지정하는 범례입니다. 이 경우 / diff_test.txt의 변경 사항은 ---로 표시되고 b / diff_test.txt의 변경 사항은 +++ 기호로 표시됩니다. 4. Diff chunks 나머지 diff 출력은 diff 'chunk'목록입니다. diff는 변경 사항이있는 파일의 섹션 만 표시합니다. 현재 시나리오에서는 간단한 시나리오로 작업하면서 하나의 청크 만 있습니다. 덩어리는 독자적인 세분화 된 출력 의미론을 가지고 있습니다. @@ -1 +1 @@ -this is a git diff test example +this is a diff example 첫 번째 줄은 청크 헤더입니다. 각 청크는 @@ 기호 안에 포함 된 헤더 앞에 추가됩니다. 헤더의 내용은 파일에 대한 변경 사항 요약입니다. 우리의 단순화 된 예에서 우리는 -1 +1 의미 라인 하나가 변경되었습니다. 보다 현실적인 diff에서는 다음과 같은 헤더가 표시됩니다. @@ -34,6 +34,8 @@ 이 머리글 예제에서는 34 행부터 6 행을 추출했습니다. 또한 34 행부터 8 행을 추가했습니다. diff chunk의 나머지 내용은 최근 변경 사항을 표시합니다. 변경된 각 줄 앞에는 + 또는 - 기호가 표시되어 변경 사항의 원본 버전을 나타냅니다. 앞에서 설명한 것처럼 -는 a / diff_test.txt의 변경 사항을 나타내며 +는 b / diff_test.txt의 변경 사항을 나타냅니다. 변경 사항 강조 표시 1. git diff --color-words git diff는 --color-words보다 훨씬 세분화 된 변경 사항을 강조하기위한 특수 모드도 제공합니다. 이 모드는 추가되거나 제거 된 행을 공백으로 토큰 화 한 다음 그 행과 비교합니다. $ git diff --color-words $ diff --git a/diff_test.txt b/diff_test.txt index 6b0c6cf..b37e70a 100644 --- a /diff_test.txt +++ b/diff_test.txt @@ -1 +1 @@ this is agit difftest example 이제 출력물에는 변경된 색으로 구분 된 단어 만 표시됩니다. 2. git diff-highlight git 소스를 복제하면 contrib이라는 하위 디렉토리가 있습니다. 여기에는 여러 git 관련 도구가 포함되어 있으며 흥미로운 비트와 조각은 아직 핵심 노드로 승격되지 않았습니다. 이 중 하나는 diff-highlight라는 Perl 스크립트입니다. diff 강조 표시는 diff 출력의 일치하는 행을 짝 지워주고 변경된 하위 단어 조각을 강조 표시합니다. $ git diff | /your/local/path/to/git-core/contrib/diff-highlight/diff-highlight diff --git a/diff_test.txt b/diff_test.txt index 6b0c6cf..b37e70a 100644 --- a /diff_test.txt +++ b/diff_test.txt @@ -1 +1 @@ -this is a git diff test example +this is a diff example 이제 우리는 가능한 가장 작은 변경 사항을 비교해 보았습니다. 바이너리 파일 비교하기 우리가 지금까지 보여준 텍스트 파일 유틸리티 외에도 git diff는 바이너리 파일에서 실행될 수 있습니다. 불행히도, 기본 출력은별로 도움이되지 않습니다. $ git diff Binary files a/script.pdf and b/script.pdf differ Git에는 diff를 수행하기 전에 이진 파일의 내용을 텍스트로 변환하는 쉘 명령을 지정할 수

2014 - 2023 CURVC Corp. All rights reserved

있는 기능이 있습니다. 그래도 약간의 설정이 필요합니다. 먼저 특정 유형의 바이너리를 텍스트로 변환하는 방법을 설명하는 textconv 필터를 지정해야합니다. 우리는 PDF 파일을 사람이 읽을 수있는 HTML로 변환하기 위해 pdftohtml이라는 간단한 유틸리티 (homebrew에서 사용 가능)를 사용하고 있습니다. 이 설정은 .git / config 파일을 편집하여 단일 저장소에 대해 설정할 수도 있고 ~ /.gitconfig를 편집하여 전역으로 설정할 수도 있습니다 [diff "pdfconv"] textconv=pdftohtml -stdout 그런 다음 하나 이상의 파일 패턴을 pdfconv 필터와 연결하기 만하면됩니다. 리포지토리의 루트에 .gitattributes 파일을 만들어이 작업을 수행 할 수 있습니다. *.pdf diff=pdfconv 일단 설정되면, git diff는 먼저 구성된 변환기 스크립트를 통해 이진 파일을 실행하고 변환기 출력을 diff합니다. jips, jars 및 기타 아카이브 : pdf2html 대신 unzip -l (또는 유사)을 사용하면 추가되거나 제거 된 경로가 표시됩니다. 이미지 커밋 : exiv2는 이미지 차원 문서와 같은 메타 데이터 변경 사항을 표시하는 데 사용할 수 있습니다. 변환 도구는 .odf, .doc 및 기타 문서 형식을 일반 텍스트로 변환하기 위해 존재합니다. 핀치 (pinch)에서 문자열은 형식 변환기가없는 바이너리 파일에서 종종 작동합니다. 파일 비교하기: git diff file git diff 명령은 명시 적 파일 경로 옵션을 전달할 수 있습니다. git diff에 파일 경로가 전달되면 diff 작업의 범위가 지정된 파일로 지정됩니다. 아래 예제는 이러한 사용법을 보여줍니다. $ git diff HEAD ./path/to/file 이 예제는 호출 될 때 ./path/to/file로 범위가 지정되며 작업 디렉토리의 특정 변경 사항을 인덱스와 비교하여 아직 준비되지 않은 변경 사항을 보여줍니다. 기본적으로 git diff는 HEAD와의 비교를 실행합니다. 위의 예에서 HEAD를 생략하면 git diff ./path/to/file과 동일한 효과가 나타납니다. $ git diff --cached ./path/to/file git diff가 --cached 옵션과 함께 호출되면 diff는 단계적 변경 사항을 로컬 저장소와 비교합니다. --cached 옵션은 --staged와 동의어입니다. 모든 변경 비교하기 git diff를 파일 경로없이 사용하면 전체 저장소의 변경 사항을 비교할 수 있습니다. 위의 파일 별 예제는 ./path/to/file 인수없이 호출 할 수 있으며 로컬 repo의 모든 파일에서 동일한 출력 결과를 갖습니다. 마지막 커밋 이후 변경 보기 기본적으로 git diff는 마지막 커밋 이후 커밋되지 않은 변경 사항을 보여줍니다. $ git diff 서로 다른 커밋의 파일 비교하기 git diff는 diff에 커밋을 전달할 수 있습니다. 몇 가지 예제 ref는 HEAD, 태그 및 분기 이름입니다. Git의 모든 커밋에는 GIT LOG를 실행할 때 얻을 수있는 커밋 ID가 있습니다. git diff에이 커밋 ID를 전달할 수도 있습니다. $ git log --prety=oneline 957fbc92b123030c389bf8b4b874522bdf2db72c add feature ce489262a1ee34340440e55a0b99ea6918e19e7a rename some classes 6b539f280d8b0ec4874671bae9c6bed80b788006 refactor some code for feature 646e7863348a427e1ed9163a9a96fa759112f102 add some copy to body $ git diff 957fbc92b123030c389bf8b4b874522bdf2db72c ce489262a1ee34340440e55a0b99ea6918e19e7a 브랜치 비교하기 두 개의 브랜치 비교하기 브랜치는 다른 모든 ref 입력과 마찬가지로 git diff와 비교됩니다. $ git diff branch1..other-feature-branch 이 예제는 도트 연산자를 소개합니다. 이 예에서 두 점은 diff 입력이 두 가지의 팁임을 나타냅니다. 점이 생략되고 분기 사이에 공백이 사용되는 경우에도 동일한 효과가 발생합니다. 또한 점 연산자 3 개가 있습니다. $ git diff branch1...other-feature-branch 세개의 도트 연산자는 첫 번째 입력 매개 변수 branch1을 변경하여 diff를 시작합니다. branch1을 두 diff 입력, 즉 branch1과 other-feature-branch의 공유 조상 사이의 공유 공통 조상 커밋의 ref로 변경합니다. 마지막 매개 변수 입력 매개 변수는 other-feature-branch의 끝으로 변경되지 않습니다. 두 브랜치의 파일 비교하기 브랜치간에 특정 파일을 비교하려면 파일 경로를 git diff의 세 번째 인수로 전달하십시오 git diff master new_branch ./diff_test.txt Git 가이드

2019-06-15 16:42:22.986

Git Commit

2014 - 2023 CURVC Corp. All rights reserved

12.png git commit 명령은 프로젝트의 현재 단계별 변경 사항에 대한 스냅 샷을 캡처합니다. 커밋 된 스냅 샷은 프로젝트의 "안전한"버전으로 생각할 수 있습니다. Git은 명시 적으로 요청하지 않으면 변경하지 않습니다. git commit을 실행하기 전에 git add 명령을 사용하여 커밋에 저장 될 프로젝트의 변경 사항을 승격 또는 '수행'합니다. 이 두 명령 git commit과 git add는 가장 자주 사용되는 두 가지 명령입니다. Git commit 과 SVN commit 차이점 동일한 이름을 공유하는 동안 git commit은 svn commit과 같은 것이 아닙니다. 이 공유 용어는 svn 배경을 가진 Git 신규 사용자에게 혼란의 포인트가 될 수 있으며 그 차이를 강조하는 것이 중요합니다. git commit 대 svn commit을 비교하는 것은 중앙 집중식 애플리케이션 모델 (svn)과 분산 애플리케이션 모델 (Git)을 비교하는 것입니다. SVN에서 커밋은 변경 사항을 로컬 SVN 클라이언트에서 원격 중앙 집중식 공유 SVN 저장소로 푸시합니다. Git에서는 저장소가 배포되고 스냅 샷은 로컬 저장소에 커밋되며 다른 Git 저장소와는 전혀 상호 작용이 필요하지 않습니다. 자식 커밋은 나중에 임의의 원격 리포지토리에 푸시 될 수 있습니다. How it works 높은 수준에서 Git은 타임 라인 관리 유틸리티로 생각할 수 있습니다. 커밋은 Git 프로젝트 타임 라인의 핵심 빌딩 블록 단위입니다. 커밋은 Git 프로젝트의 타임 라인을 따라 스냅 샷 또는 마일스톤으로 생각할 수 있습니다. 커밋은 git commit 명령으로 생성되어 해당 시점의 프로젝트 상태를 캡처합니다. 스냅 샷은 항상 로컬 저장소에 커밋됩니다. 이는 SVN과 근본적으로 다르며 작업 카피는 중앙 저장소에 커밋됩니다. 대조적으로, 힘내라는 준비가 끝날 때까지 중앙 저장소와 상호 작용하도록 강요하지 않습니다. 준비 영역은 작업 디렉토리와 프로젝트 기록 사이의 버퍼이므로 각 개발자의 로컬 저장소는 기여와 중앙 저장소 사이의 버퍼입니다. Git 사용자를위한 기본 개발 모델을 변경합니다. Git 개발자는 변경 사항을 중앙 repo에 직접 커밋하는 대신 로컬 레포에 커밋을 축적 할 수 있습니다. 이것은 SVN 스타일의 협업에 비해 많은 이점을 가지고 있습니다. 즉, 기능을 원자 단위로 분할하고, 관련 커밋을 그룹화하고, 중앙 저장소에 공개하기 전에 로컬 내역을 정리하는 것을 더 쉽게 만듭니다. 또한 개발자는 격리 된 환경에서 작업 할 수 있으므로 다른 사용자와 편리하게 병합 할 때까지 통합이 지연됩니다. 격리 및 지연된 통합은 개별적으로 유익하지만, 자주 및 작은 단위로 통합하는 것이 팀의 이익입니다. 힘내 팀 공동 작업에 대한 우수 사례에 대한 자세한 내용은 팀에서 어떻게 힘내 워크 플로우를 구성하는지 읽어보십시오. Snapshots, not differences SVN과 Git 사이의 실질적인 차이점을 제외하고는 기본 구현은 완전히 다른 디자인 철학을 따릅니다. SVN이 파일의 차이를 추적하는 반면 Git의 버전 제어 모델은 스냅 샷을 기반으로합니다. 예를 들어, SVN 커밋은 저장소에 추가 된 원본 파일과 비교되는 diff로 구성됩니다. Git은 모든 커밋에서 각 파일의 전체 내용을 기록한다. 14-1.png 이는 특정 버전의 파일을 diff에서 "어셈블"할 필요가 없기 때문에 많은 Git 작업을 SVN보다 훨씬 빠르게 만듭니다. 각 파일의 전체 버전은 Git의 내부 데이터베이스에서 즉시 사용할 수 있습니다. Git의 스냅 샷 모델은 버전 제어 모델의 거의 모든 측면에 광범위한 영향을 미치며 분기 및 병합 도구에서부터 협업 작업 흐름에 이르는 모든 것에 영향을 미칩니다. 공통 옵션 git commit 준비된 스냅 샷을 커밋합니다. 그러면 커밋 메시지를 묻는 텍스트 편집기가 시작됩니다. 메시지를 입력 한 후 파일을 저장하고 편집기를 닫아 실제 커밋을 만듭니다. git commit -a 작업 디렉토리의 모든 변경 사항에 대한 스냅 샷을 커밋합니다. 이것은 오직 추적 된 파일 (히스토리의 어느 시점에서 git add로 추가 된 것들)에 대한 수정 만 포함합니다. git commit -m "commit message" 전달 된 커밋 메시지로 즉시 커밋을 만드는 바로 가기 명령입니다. 기본적으로 git commit은 로컬로 구성된 텍스트 편집기를 열고 커밋 메시지를 입력하라는 메시지를 표시합니다. -m 옵션을 전달하면 인라인 메시지를 선호하는 텍스트 편집기 프롬프트가 표시되지 않습니다. git commit -am "commit message" -a 및 -m 옵션을 결합한 파워 유저 바로 가기 명령입니다. 이 조합은 즉시 모든 단계적 변경 사항의 커밋을 생성하고 인라인 커밋 메시지를받습니다. git commit --amend 이 옵션은 커밋 명령에 다른 수준의 기능을 추가합니다. 이 옵션을 전달하면 마지막 커밋이 수정됩니다. 새 커밋을 만드는 대신 이전 커밋에 단계별 변경 사항이 추가됩니다. 이 명령은 시스템에 구성된 텍스트 편집기를 열고 이전에 지정한 커밋 메시지를 변경하라는 메시지를 표시합니다. Examples 커밋으로 변경 저장하기 다음 예제는 현재 브랜치에서 hello.py라는 파일의 일부 내용을 편집 한 것으로 가정하고 프로젝트 히스토리에 커밋 할 준비가 되어 있다고 가정합니다. 먼저 git add로 파일을 준비한 다음 준비된 스냅 샷을 커밋할 수 있습니다. git add hello.py 이 명령은 hello.py를 Git 스테이징 영역에 추가합니다. git status 명령을 사용하여이 작업의 결과를 검사 할 수 있습니다. git status On branch master Changes to be committed: (use "git reset HEAD <file>..." to unstage) new file: hello.py 녹색 출력 새로운 파일 : hello.py는 hello.py가 다음 커밋과 함께 저장 될 것임을 나타냅니다. 커밋에서 다음을 실행하여 만듭니다. git commit 커밋 된 로그 메시지를 요구하는 텍스트 편집기 (git config를 통해 사용자 정의 가능)가 열립니다. # Please enter the commit message for your changes. Lines starting # with '#' will be ignored, and an empty message aborts the commit. # On branch master # Changes to be committed: # (use "git reset HEAD ..." to unstage) # #modified: hello.py Git은 특정 형식 제약 조건을 따르는 커밋 메시지를 요구하지 않지만 표준 형식은 첫 줄에 전체 커밋을 50 자 미만으로 요약하고 빈 줄을 남기고 변경된 내용에 대한 자세한 설명을 제공합니다. 예 : Change the message displayed by hello.py - Update the sayHello() function to output the user's name - Change the sayGoodbye() function to a friendlier message 전자 메일과 마찬가지로 커밋 메시지의 첫 번째 줄을 제목 줄로 사용하는 것이 일반적입니다. 나머지 로그 메시지는 본문으로 간주되어 커밋 변경 집합의 세부 정보를 전달하는 데 사용됩니다. 많은 개발자들이 커밋 메시지에 현재 시제를 사용하기를 좋아한다는 점에 유의하십시오. 이것은 리포지토리에서의 작업과 같이 더 많이 읽을 수있게 해줌으로써 많은 역사 작성 작업을 더욱 직관적으로 만듭니다. 마지막 커밋 수정하기 위의 hello.py 예제를 계속 진행하십시오. hello.py에 대한 추가 업데이트를하고 다음을 실행합니다. git add hello.py git commit --amend 그러면 다시 구성된 텍스트 편집기가 열립니다. 그러나 이번에는 이전에 입력 한 커밋 메시지로 미리 채워질 것입니다. 이것은 우리가 새로운 커밋을 생성하는 것이 아니라 마지막 커밋을 편집하고 있음을 나타냅니다. Git 가이드

2014 - 2023 CURVC Corp. All rights reserved

2019-06-15 19:41:44.771

.gitignore

17.png Git은 작업 카피의 모든 파일을 세 가지 중 하나로 본다. 추적 - 이전에 수행되었거나 확약 된 파일. untracked - 준비되거나 커밋되지 않은 파일. 또는 무시 됨 - Git이 명시 적으로 무시하도록 지시 한 파일. 무시 된 파일은 일반적으로 저장소 소스에서 파생되거나 커밋되지 않아야하는 빌드 아티팩트 및 컴퓨터 생성 파일입니다. 몇 가지 일반적인 예는 다음과 같습니다. / node_modules 또는 / 패키지의 내용과 같은 종속성 캐시 .o, .pyc 및 .class 파일과 같은 컴파일 된 코드 / bin, / out 또는 / target과 같은 출력 디렉토리 빌드 .log, .lock 또는 .tmp와 같이 런타임에 생성되는 파일 .DS_Store 또는 Thumbs.db와 같은 숨겨진 시스템 파일 .idea / workspace.xml과 같은 개인 IDE 구성 파일 무시 된 파일은 저장소의 루트에서 체크인 된 .gitignore라는 특수 파일에서 추적됩니다. 명시 적으로 git ignore 명령은 없습니다. 대신에 .gitignore 파일을 편집하고 커밋해야합니다. 무시할 새 파일이 있으면 수동으로 커밋해야합니다. .gitignore 파일은 저장소의 파일 이름과 일치하는 패턴을 포함하여 무시해야하는지 여부를 결정합니다. Git ignore 패턴 .gitignore는 globbing 패턴을 사용하여 파일 이름과 일치시킵니다. 다양한 기호를 사용하여 패턴을 구성 할 수 있습니다. Pattern Example matches Explanation* **/logs logs/debug.log logs/monday/foo.bar build/logs/debug.log You can prepend a pattern with a double asterisk to match directories anywhere in the repository. **/logs/debug.log logs/debug.log build/logs/debug.log but not logs/build/debug.log You can also use a double asterisk to match files based on their name and the name of their parent directory. *.log debug.log foo.log .log logs/debug.log An asterisk is a wildcard that matches zero or more characters. *.log !important.log debug.log trace.log but not important.log logs/important.log Prepending an exclamation mark to a pattern negates it. If a file matches a pattern, but alsomatches a negating pattern defined later in the file, it will not be ignored. *.log !important/*.log trace.* debug.log important/trace.log but not important/debug.log Patterns defined after a negating pattern will re-ignore any previously negated files. /debug.log debug.log but not logs/debug.log Prepending a slash matches files only in the repository root. debug.log debug.log logs/debug.log By default, patterns match files in any directory debug?.log debug0.log debugg.log but not debug10.log A question mark matches exactly one character. debug[0-9].log debug0.log debug1.log but not debug10.log Square brackets can also be used to match a single character from a specified range. debug[01].log debug0.log debug1.log but not debug2.log debug01.log Square brackets match a single character form the specified set. debug[!01].log debug2.log but not debug0.log debug1.log debug01.log An exclamation mark can be used to match any character except one from the specified set. debug[a-z].log debuga.log debugb.log but not debug1.log Ranges can be numeric or alphabetic. logs logs logs/debug.log logs/latest/foo.bar build/logs build/logs/debug.log If you don't append a slash, the pattern will match both files and the contents of directories with that name. In the example matches on the left, both directories and files named logs are ignored logs/ logs/debug.log logs/latest/foo.bar build/logs/foo.bar build/logs/latest/debug.log Appending a slash indicates the pattern is a directory. The entire contents of any directory in the repository matching that name - including all of its files and subdirectories - will be ignored logs/ !logs/important.log logs/debug.log logs/important.log Wait a minute! Shouldn't logs/important.logbe negated in the example on the left Nope! Due to a performance-related quirk in Git, you can not negate a file that is ignored due to a pattern matching a directory logs/**/debug.log logs/debug.log logs/monday/debug.log logs/monday/pm/debug.log A double asterisk matches zero or more directories. logs/*day/debug.log logs/monday/debug.log logs/tuesday/debug.log but not logs/latest/debug.log Wildcards can be used in directory names as well. logs/debug.log logs/debug.log but not debug.log build/logs/debug.log Patterns specifying a file in a particular directory are relative to the repository root. (You can prepend a slash if you like, but it doesn't do anything special.) **이 설명은 .gitignore 파일이 협약과 마찬가지로 저장소의 최상위 디렉토리에 있다고 가정합니다. 저장소에 .gitignore 파일이 여러 개있는 경우 "저장소 루트"를 ".gitignore 파일이있는 디렉토리"로 정신적으로 대체하십시오 (팀의 온전한 고려를 위해 통합을 고려하십시오). * 이러한 문자 외에도 .gitignore 파일에 #을 사용하여 주석을 포함 할 수 있습니다. # ignore all logs *.log 파일이나 디렉토리가있는 경우 ₩를 사용하여 .gitignore 패턴 문자를 이스케이프 처리 할 수 있습니다. # ignore the file literally named foo[01].txt foo₩[01₩].txt 저장소에 .gitignore 공유하기 Git 무시 규칙은 일반적으로 저장소의 루트에있는 .gitignore 파일에 정의되어 있습니다. 그러나 저장소의 여러 디렉토리에 여러 .gitignore 파일을 정의하도록 선택할 수 있습니다. 특정 .gitignore 파일의 각 패턴은 해당 파일이 들어있는 디렉토리를 기준으로 테스트됩니다. 그러나 규칙과 가장 단순한 접근법은 루트에 단일 .gitignore 파일을 정의하는 것입니다. .gitignore 파일이 체크인되면 리포지토리의 다른 파일과 마찬가지로 버전이 지정되고 푸시 할 때 팀원과 공유됩니다. 일반적으로 .gitignore에는 저장소의 다른 사용자에게 도움이되는 패턴 만 포함시켜야합니다. Personal Git ignore rules 특정 저장소에 대한 개인 무시 패턴을 .git / info / exclude의 특수 파일에 정의 할 수도 있습니다. 이러한 버전은 버전이 지정되지 않으며 저장소와 함께 배포되지 않으므로 이점 만 얻을 수있는 패턴을 포함하는 것이 좋습니다. 예를 들어 사용자 정의 로깅 설정이나 저장소의 작업 디렉토리에 파일을 생성하

2014 - 2023 CURVC Corp. All rights reserved

는 특수 개발 도구가있는 경우 .git / info / exclude에 추가하여 실수로 저장소에 커밋되지 않도록 할 수 있습니다. Global Git ignore rules 또한 Git core.excludesFile 속성을 설정하여 로컬 시스템의 모든 리포지토리에 대한 전역 Git 무시 패턴을 정의 할 수 있습니다. 이 파일을 직접 만들어야합니다. 전역 .gitignore 파일을 어디에 둘지 확실치 않으면 홈 디렉토리가 나쁜 선택이 아니며 나중에 쉽게 찾을 수 있습니다. 파일을 만들었 으면 git config로 위치를 구성해야합니다. $ touch ~/.gitignore $ git config --global core.excludesFile ~/.gitignore 서로 다른 파일 유형이 서로 다른 프로젝트와 관련되므로 어떤 패턴을 전역 적으로 무시할지 선택해야합니다. 일부 개발자 도구로 만든 특별 운영 체제 파일 (예 : .DS_Store 및 thumbs.db) 또는 임시 파일은 전 세계적으로 무시할 수있는 대표적인 후보입니다. Ignoring a previously committed file 과거에 커밋 한 파일을 무시하려면 저장소에서 파일을 삭제 한 다음 .gitignore 규칙을 추가해야합니다. git rm과 함께 --cached 옵션을 사용하면 파일이 저장소에서 삭제되지만 작업 디렉토리에는 무시 된 파일로 남아있게됩니다. $ echo debug.log >> .gitignore $ git rm --cached debug.log rm 'debug.log' $ git commit -m "Start ignoring debug.log" 저장소 및 로컬 파일 시스템에서 파일을 삭제하려면 --cached 옵션을 생략 할 수 있습니다. Committing an ignored file git add와 함께 -f (또는 --force) 옵션을 사용하여 무시 된 파일을 저장소에 강제로 적용 할 수 있습니다. $ cat .gitignore *.log $ git add -f debug.log $ git commit -m "Force adding debug.log" 일반적인 패턴 (예 : * .log)이 정의되어 있지만 특정 파일을 커미트하려는 경우이 작업을 고려할 수 있습니다. 그러나 더 좋은 해결책은 일반적인 규칙에 대한 예외를 정의하는 것입니다. $ echo !debug.log >> .gitignore $ cat .gitignore *.log !debug.log $ git add debug.log $ git commit -m "Adding debug.log" 이 접근법은 팀원에게보다 분명하고 혼란스럽지 않습니다. Stashing an ignored file git stash는 로컬 변경 사항을 일시적으로 저장 및 되돌리기위한 강력한 Git 기능으로, 나중에 다시 적용 할 수 있습니다. 예상대로, 기본적으로 git stash는 무시 된 파일을 무시하고 Git에 의해 추적되는 파일에 대한 변경 사항 만 숨 깁니다. 그러나, --all 옵션을 사용하여 git stash를 호출하면 무시되거나 변경되지 않은 파일에 변경 사항을 숨길 수 있습니다. Debugging .gitignore files 복잡한 .gitignore 패턴이 있거나 여러 .gitignore 파일에 패턴이 퍼져있는 경우 특정 파일을 왜 무시하는지 추적하기가 어려울 수 있습니다. git check-ignore 명령을 -v (또는 --verbose) 옵션과 함께 사용하여 어떤 패턴이 특정 파일을 무시하게하는지 결정할 수 있습니다. $ git check-ignore -v debug.log .gitignore:3:*.log debug.log 결과: <file containing the pattern> : <line number of the pattern> : <pattern> <file name> 원하는 경우 여러 파일 이름을 git check-ignore에 전달할 수 있으며 이름 자체는 저장소에있는 파일에 해당 할 필요조차 없습니다. Git 가이드

2019-06-15 18:41:23.002

Git Stash

작업 복사본에 대한 숨김 보관함 (또는 숨김) 변경 내용을 숨겨서 다른 작업을 할 수있게 한 다음 나중에 다시 적용하여 나중에 다시 적용 할 수 있습니다. 숨기기는 컨텍스트를 신속하게 전환하고 다른 작업을해야하는 경우 편리하지만 코드를 변경하는 중반에 있으며 커밋 할 준비가되지 않았습니다. 작업 숨기기 git stash 명령은 커밋되지 않은 변경 (준비 및 언 스테이지 모두)을 취하고 나중에 사용하기 위해 저장 한 다음 작업 복사본에서 되돌립니다. 예 : $ git status On branch master Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html $ git stash Saved working directory and index state WIP on master: 5002d47 our new homepage HEAD is now at 5002d47 our new homepage $ git status On branch master nothing to commit, working tree clean 이 시점에서 여러분은 자유롭게 변경을 할 수 있고, 새로운 커밋을 만들고, 분기를 전환하고, 다른 Git 작업을 수행 할 수 있습니다. 그런 다음 다시 돌아와서 준비가되었을 때 은닉을 다시 적용하십시오. 숨겨진 저장소는 Git 저장소에 로컬로 저장됩니다. 밀어 넣을 때 은폐는 서버로 전송되지 않습니다. 숨겨진 변경 적용하기 git stash pop으로 이전에 숨긴 변경 사항을 다시 적용 할 수 있습니다. $ git status On branch master nothing to commit, working tree clean $ git stash pop On branch master Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html Dropped refs/stash@{0} (32b3aa1d185dfe6d57b3c3cc3b32cbf3e380cc6a) Popping your stash removes the changes from your stash and reapplies them to your working copy. Alternatively, you can reapply the changes to your working copy and keep them in your stash with git stash apply: $ git stash apply On branch master Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html 여러 분기에 동일한 숨겨진 변경 사항을 적용하려는 경우 유용합니다. stashing의 기본 사항을 알았으므로 git stash를주의해야합니다. 기본적으로 Git은 추적되지 않거나 무시되는 파일의 변경 사항을 숨기지 않습니다. untracked or ignored 파일 숨기기 기본적으로 git stash를 실행하면 다음 항목이 숨겨집니다. 색인에 추가 된 변경 사항 (단계별 변경 사항) 현재 Git에 의해 추적되는 파일에 대한 변경 사항 (unstaged changes) 그러나 다음 항목은 숨겨지지 않을 것입니다 : 작업 복사본에서 아직 준비되지 않은 새 파일 무시 된 파일 위 예제에 세 번째 파일을 추가했지만 스테이지를 만들지 않으면 (즉, git add를 실행하지 않음) git stash가 stash하지 않습니다. $ script.js $ git status On branch master Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html Untracked files: script.js $ git stash Saved working directory and index state WIP on master: 5002d47 our new

2014 - 2023 CURVC Corp. All rights reserved

homepage HEAD is now at 5002d47 our new homepage $ git status On branch master Untracked files: script.js -u 옵션 (또는 --include-untracked)을 추가하면 git stash가 untracked 파일을 숨긴다. $ git status On branch master Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html Untracked files: script.js $ git stash -u Saved working directory and index state WIP on master: 5002d47 our new homepage HEAD is now at 5002d47 our new homepage $ git status On branch master nothing to commit, working tree clean git stash를 실행할 때 -a 옵션 (또는 --all)을 전달하여 무시 된 파일에 대한 변경 사항을 포함 할 수 있습니다. 15.png 숨겨진 목록 (stash) 관리하기 git stash를 여러 번 실행하여 숨겨진 목록을 만든 다음 git stash list를 사용하여 볼 수 있습니다. 기본적으로 stash는 작성된 분기 또는 커밋 위의 "진행중인 작업"(WIP)으로 식별 됩니다. 잠시 후 각 은닉 내용을 기억하는 것은 어려울 수 있습니다. $ git stash list stash@{0}: WIP on master: 5002d47 our new homepage stash@{1}: WIP on master: 5002d47 our new homepage stash@{2}: WIP on master: 5002d47 our new homepage 좀 더 많은 문맥을 제공하기 위해 stash에 설명을 추가하고 git stash save "message"를 사용하여 주석을 달아주는 것이 좋습니다. $ git stash save "add style to our site" Saved working directory and index state On master: add style to our site HEAD is now at 5002d47 our new homepage $ git stash list stash@{0}: On master: add style to our site stash@{1}: WIP on master: 5002d47 our new homepage stash@{2}: WIP on master: 5002d47 our new homepage 기본적으로 git stash pop은 가장 최근에 생성 된 stash를 다시 적용합니다 : stash @ {0} 식별자를 마지막 인수로 전달하여 다시 적용 할 스태프를 선택할 수 있습니다 (예 : $ git stash pop stash@{2} 숨겨진 내용 비교 (stash diffs) git stash show를 사용하여 stash 요약을 볼 수 있습니다. $ git stash show index.html | 1 + style.css | 3 +++ 2 files changed, 4 insertions(+) 또는 stash 전체 diff를 보려면 -p 옵션 (또는 - patch)을 전달하십시오. $ git stash show -p diff --git a/style.css b/style.css new file mode 100644 index 0000000..d92368b --- /dev/null +++ b/style.css @@ -0,0 +1,3 @@ +* { + text-decoration: blink; +} diff --git a/index.html b/index.html index 9daeafb..ebdcbd2 100644 --- a/index.html +++ b/index.html @@ -1 +1,2 @@ +<link rel="stylesheet" href="style.css"/> 부분 stashes 단일 파일, 파일 모음 또는 파일 내 개별 변경 사항을 숨기도록 선택할 수도 있습니다. git stash에 -p 옵션 (또는 --patch)을 전달하면 작업 복사본에서 변경된 각 "덩어리"를 반복하여 숨길 지 여부를 묻습니다. $ git stash -p diff --git a/style.css b/style.css new file mode 100644 index 0000000..d92368b --- /dev/null +++ b/style.css @@ -0,0 +1,3 @@ +* { + text-decoration: blink; +} Stash this hunk [y,n,q,a,d,/,e,?]? y diff --git a/index.html b/index.html index 9daeafb..ebdcbd2 100644 --- a/index.html +++ b/index.html @@ -1 +1,2 @@ +<link rel="stylesheet" href="style.css"/> Stash this hunk [y,n,q,a,d,/,e,?]? n 15-2.png 일반적으로 유용한 것들은 다음과 같습니다 : Command Description / search for a hunk by regex ? help n don't stash this hunk q quit (any hunks that have already been selected will be stashed) s split this hunk into smaller hunks y stash this hunk 명시 적 "중단"명령은 없지만 CTRL-C (SIGINT)를 누르면 숨김 프로세스가 중단됩니다. Stash로부터 브랜치 생성하기 브랜치의 변경사항이 stash의 변경사항과 다른 경우 stash를 적용 할 때 충돌이 발생할 수 있습니다. 대신 git stash branch를 사용하여 숨겨진 변경 사항을 적용 할 새 브랜치를 만들 수 있습니다 : $ git stash branch add-stylesheet stash@{1} Switched to a new branch 'add-stylesheet' On branch add-stylesheet Changes to be committed: new file: style.css Changes not staged for commit: modified: index.html Dropped refs/stash@{1} (32b3aa1d185dfe6d57b3c3cc3b32cbf3e380cc6a) 이것은 당신이 숨긴 곳을 만든 커밋을 기반으로 새로운 브랜치를 체크 아웃하고 숨겨진 변경 사항을 그 위에 놓습니다. stash 비우기 더 이상 stash가 필요 없다고 판단되면 git stash drop으로 삭제할 수 있습니다. $ git stash drop stash@{1} Dropped stash@{1} (17e2697fd8251df6163117cb3d58c1f62a5e7cdb) 또는 다음과 같이 모든 숨기기를 삭제할 수 있습니다. $ git stash clear How git stash works Stash가 어떻게 동작하는지 궁금하다면 이 절을 읽어도 좋습니다. 단지 stash 사용 방법이 알고싶다면 이 절을 읽지 않아도 됩니다. stash는 저장소에서 실제로 커밋 객체로 인코딩됩니다. .git / refs / stash의 특별한 ref는 가장 최근에 생성 된 stash를 가리키며, 이전에 생성 된 stash는 stash ref의 reflog에 의해 참조됩니다. 이것이 당신이 stash @ {n}을 은닉하여 참조하는 이유입니다. 실제로는 stash ref의 n 번째 reflog 항목을 참조하고 있습니다. stash는 커밋 일 뿐이므로 git log로 검사 할 수있습니다. $ git log --oneline --graph stash@{0} *-. 953ddde WIP on master: 5002d47 our new homepage |₩ ₩ | | * 24b35a1 untracked files on master: 5002d47 our new homepage | * 7023dd4 index on master: 5002d47 our new homepage |/ * 5002d47 our new homepage 숨긴 것에 따라, 하나의 git stash 연산은 2 ~ 3 개의 새로운 커밋을 생성합니다. 위 다이어그램의 커밋은 다음과 같습니다. stash @ {0}, git stash를 실행할 때 작업 복사본에 있던 추적 된 파일을 저장하는 새로운 커밋 @ {0} 님의 첫 번째 부모를 숨기고 git stash를 실행할 때 HEAD에 있던 기존 커밋을 숨긴다. @ {0}의 두 번째 부모를 숨기고, git stash를 실행할 때 색인을 나타내는 새로운 커밋 @ {0}의 세 번째 부모를 숨기고, git stash를 실행할 때 작업 복사본에 있던 untracked 파일을 나타내는 새로운 커밋. 이 세 번째 상위 항목은 다음 경우에만 생성됩니다. 작업 복사본에는 실제로 untracked 파일이 포함되어 있습니다. 과 git stash를 호출 할 때 --include-untracked 또는 --all 옵션을 지정했습니다. git stash가 작업 트리와 인덱스를 커밋으로 인코딩하는 방법은 다음과 같습니다. 숨겨지기 전에 작업 트리에 추적 된 파일, 추적 할 수없는 파일 및 무시 된 파일에 대한 변경 사항이 포함될 수 있습니다. 이러한 변경 사항 중 일부는 색인에 포함될 수도 있습니다. 16.png git stash를 호출하면 추적 된 파일에 대한 모든 변경 사항이 DAG에서 두 개의 새로운 커밋으로 인코딩됩니다. 하나는 무단 변경에 대한 것이고 다른 하나는 인덱스에 준비된 변경에 대한 것입니다. 특수한 refs / stash ref가 그들을 가리 키도록 업데이트됩니다. 16-1.png --include-untracked 옵션을 사용하면 추적되지 않은 파일에 대한 변경 사항을 추가 커밋으로 인코딩 할 수 있습니다. 16-2.png --all 옵션을 사용하면 동일한 커밋에서 추적되지 않은 파일의 변경 사항과 함께 무시 된 파일의 변경 사항이 포함됩니다. 16-3.png git stash pop을 실행하면 위 커밋의 변경 사항이 작업 복사본과 인덱스를 업데이트하는 데 사용되며 stash reflog는 셔플되어 팝 된 커

2014 - 2023 CURVC Corp. All rights reserved

밋을 제거합니다. 팝 된 커밋은 즉시 삭제되지 않지만 향후 가비지 수집을위한 후보가 됩니다. Git 가이드

2019-06-15 16:27:22.431

Git Add

git add 명령은 작업 디렉토리의 변경 사항을 스테이징 영역에 추가합니다. Git에게 다음 커밋에서 특정 파일에 대한 업데이트를 포함하길 원한다는 것을 알려줍니다. 그러나 git add는 저장소에 중요한 영향을 미치지 않습니다. 변경 사항은 git commit을 실행할 때까지 실제로 기록되지 않습니다. 이 명령들과 함께 작업 디렉토리와 스테이징 영역의 상태를 보려면 git 상태가 필요합니다. How it works git add와 git commit 명령은 근본적인 Git 작업 흐름을 구성합니다. 이 명령은 팀의 공동 작업 모델에 관계없이 모든 Git 사용자가 이해해야하는 두 가지 명령입니다. 그것들은 저장소의 히스토리에 프로젝트 버전을 기록하는 수단이다. 프로젝트를 개발하는 것은 기본적인 편집 / 단계 / 커밋 패턴을 중심으로 이루어집니다. 먼저 작업 디렉토리에서 파일을 편집합니다. 프로젝트의 현재 상태 사본을 저장할 준비가되면 git add 명령으로 변경 사항을 저장합니다. 준비된 스냅 샷에 만족하면 git commit을 사용하여 프로젝트 히스토리에 커밋합니다. git reset 명령은 커밋 또는 준비된 스냅 샷을 실행 취소하는 데 사용됩니다. git add 및 git commit 외에도 세 번째 명령 인 git push는 완전한 협업 작업 흐름을 위해 필수적입니다. git push는 협업을 위해 커밋 된 변경 사항을 원격 저장소에 보내는 데 사용됩니다. 이렇게하면 다른 팀 구성원이 저장된 변경 사항 집합에 액세스 할 수 있습니다. 11.png git add 명령을 저장소에 파일을 추가하는 svn add와 혼동해서는 안됩니다. 대신, git add는 더 추상적 인 변경 레벨에서 작동합니다. 즉, 파일을 변경할 때마다 git add를 호출해야하지만 svn add는 각 파일에 대해 한 번만 호출하면됩니다. 중복되는 것처럼 들릴 수도 있지만,이 워크 플로우는 프로젝트를 체계적으로 정리하는 것을 훨씬 쉽게 만듭니다. The staging area git add 명령의 주요 기능은 작업 디렉토리에서 보류중인 변경 사항을 git staging 영역으로 승격시키는 것입니다. 준비 영역은 Git의 고유 한 기능 중 하나이며, SVN (또는 심지어 Mercurial) 배경에서 오는 경우에는 머리를 감싸는 데 약간의 시간이 걸릴 수 있습니다. 작업 디렉토리와 프로젝트 기록 사이의 버퍼로 생각하는 것이 도움이됩니다. 준비 영역은 Git의 "세 나무"중 하나, 작업 디렉토리 및 커밋 기록 중 하나로 간주됩니다. 마지막 커밋 이후에 변경 한 사항을 모두 커밋하는 대신 스테이지에서 관련 변경 사항을 포커스가있는 스냅 샷으로 그룹화하여 실제로 프로젝트 기록에 커밋 할 수 있습니다. 즉, 무관 한 파일에 대한 모든 종류의 편집 작업을 수행 한 다음 관련 변경 사항을 스테이지에 추가하고이를 논리적 커밋으로 분리하여 개별적으로 커밋 할 수 있습니다. 모든 개정 관리 시스템에서와 마찬가지로 원자 커밋을 만들어 버그를 추적하고 변경 사항을 프로젝트의 나머지 부분에 미치는 영향을 최소화하면서 쉽게 되돌릴 수 있도록하는 것이 중요합니다. 공통 옵션 git add <file> 다음 커밋을 위해 <file>의 모든 변경 사항을 준비하십시오. git add <directory> 다음 커밋을 위해 <directory>의 모든 변경 사항을 스테이지하십시오. git add -p 다음 커밋에 추가 할 파일의 부분을 선택할 수있는 대화 형 스테이징 세션을 시작하십시오. 이렇게하면 변경 사항이 표시되고 명령을 묻는 메시지가 나타납니다. 청크를 스테이지하려면 y를, 청크를 무시하려면, 작은 청크로 분할하고, 청크를 수동으로 편집하고, q를 종료하십시오. Examples 새 프로젝트를 시작할 때 git add는 svn import와 동일한 기능을 제공합니다. 현재 디렉토리의 초기 확약을 작성하려면 다음 두 명령을 사용하십시오. git add . git commit 프로젝트를 시작하고 실행하면 git add에 경로를 전달하여 새 파일을 추가 할 수 있습니다. git add hello.py git commit 위의 명령을 사용하여 기존 파일의 변경 사항을 기록 할 수도 있습니다. 다시 한번 말하지만, Git은 새로운 파일의 준비 변경과 이미 저장소에 추가 된 파일의 변경을 구분하지 않습니다. Git 가이드

2019-06-16 06:14:06.0

Git Log

2014 - 2023 CURVC Corp. All rights reserved

git log 명령은 커밋 된 스냅 샷을 표시합니다. 프로젝트 기록을 나열하고, 필터링하고, 특정 변경 사항을 검색 할 수 있습니다. git status를 사용하면 작업 디렉토리와 스테이징 영역을 검사 할 수 있지만 git log는 커밋 된 히스토리에서만 작동합니다. 18.png 로그 출력은 커밋을 필터링하는 것에서부터 완전히 사용자 정의 된 형식으로 표시하는 것까지 여러 가지 방법으로 사용자 정의 할 수 있습니다. git log의 가장 일반적인 설정 중 일부는 다음과 같습니다. 사용법 git log 기본 포맷을 사용하여 전체 커밋 기록을 표시하십시오. 출력이 둘 이상의 화면을 차지하는 경우, 스페이스를 사용하여 스크롤하고 q를 사용하여 종료 할 수 있습니다. git log -n <limit> 커밋 수를 <limit>만큼 제한하십시오. 예를 들어, git log -n 3은 3 개의 커밋 만 표시합니다. git log --oneline 각 커밋을 한 줄로 압축하십시오. 이것은 프로젝트 히스토리에 대한 높은 수준의 개요를 얻는 데 유용합니다. git log --stat 일반 자식 로그 정보와 함께 변경된 파일과 각 파일에 추가 또는 삭제 된 행의 상대 번호가 포함됩니다. git log -p 각 커밋을 나타내는 패치를 표시합니다. 이것은 각 커밋의 전체 diff를 보여줍니다. 이것은 프로젝트 히스토리에 대해 가장 자세히 볼 수 있습니다. git log --author="<pattern>" 특정 작성자가 커밋을 검색합니다. <pattern> 인수는 일반 문자열이나 정규 표현식이 될 수 있습니다. git log --grep="<pattern>" <pattern>과 일치하는 커밋 메시지로 커밋을 검색합니다.이 커밋 메시지는 일반 문자열이나 정규 표현식이 될 수 있습니다. git log <since>..<until> <since>와 <until> 사이에 발생하는 커밋 만 표시합니다. 두 인수 모두 커밋 ID, 분기 이름, HEAD 또는 다른 종류의 수정 참조 일 수 있습니다. git log <file> 지정된 파일을 포함하는 커밋 만 표시하십시오. 이것은 특정 파일의 기록을 쉽게 볼 수있는 방법입니다. git log --graph --decorate --oneline 몇 가지 유용한 옵션을 고려해야합니다. 커밋 메시지의 왼쪽에 커밋의 텍스트 기반 그래프를 그리는 --graph 플래그. --decorate는 표시된 커밋의 브랜치 또는 태그 이름을 추가합니다. --online은 커밋 정보를 한 줄에 표시하여 커밋을 한 눈에 쉽게 탐색 할 수있게합니다. Git 가이드

2019-06-16 06:52:01.494

Git Blame

tag.png git blame 명령은 광범위한 사용법 옵션이있는 다목적 문제 해결 유틸리티입니다. git blame의 높은 수준의 기능은 파일의 특정 커밋 된 행에 첨부 된 작성자 메타 데이터의 표시입니다. 이는 파일 히스토리의 특정 지점을 검사하고 최종 작성자가 행을 수정 한 사람에 대한 컨텍스트를 얻는 데 사용됩니다. 이 코드는 특정 코드의 내역을 탐색하고 코드가 저장소에 추가 된 대상, 방법 및 이유를 알아내는 데 사용됩니다. Git blame은 종종 GUI 디스플레이와 함께 사용됩니다. Bitbucket과 같은 온라인 Git 호스팅 사이트는 저장소에 대한 blame 뷰를 제공합니다. 이러한 뷰는 pull rquest 및 커밋에 대한 협업에서 참조됩니다. 또한, Git 통합이있는 대부분의 IDE에는 동적 인 blame 기능을 제공합니다. 작동 원리 git blame을 증명하기 위해서 우리는 약간의 역사가있는 저장소가 필요하다. 우리는 오픈 소스 프로젝트 인 git-blame-example을 사용할 것입니다. 이 오픈 소스 프로젝트는 다른 저자의 커밋이있는 README.md http://README.md 파일이 들어있는 간단한 저장소입니다. git blame 사용 예제의 첫 번째 단계는 예제 저장소를 복제하는 것입니다. git clone https://kevzettler@bitbucket.org/kevzettler/git-blame-example.git && cd git-blame-example 이제 예제 코드의 복사본을 얻었으므로 git blame으로 살펴보기 시작할 수있다. 예제 repo의 상태는 git log를 사용하여 검사 할 수 있습니다. 커밋 내역은 다음과 같아야합니다. $ git log commit 548dabed82e4e5f3734c219d5a742b1c259926b2 Author: Juni Mukherjee <jmukherjee@atlassian.com> Date: Thu Mar 1 19:55:15 2018 +0000 Another commit to help git blame track the who, the what, and the when commit eb06faedb1fdd159d62e4438fc8dbe9c9fe0728b Author: Juni Mukherjee <jmukherjee@atlassian.com> Date: Thu Mar 1 19:53:23 2018 +0000 Creating the third commit, along with Kev and Albert, so that Kev can get git blame docs. commit 990c2b6a84464fee153253dbf02e845a4db372bb Merge: 82496ea 89feb84 Author: Albert So <aso@atlassian.com> Date: Thu Mar 1 05:33:01 2018 +0000 Merged in albert-so/git-blame-example/albert-so/readmemd-edited-online-with-bitbucket-1519865641474 (pull request #2) README.md edited online with Bitbucket commit 89feb84d885fe33d1182f2112885c2a64a4206ec Author: Albert So <aso@atlassian.com> Date: Thu Mar 1 00:54:03 2018 +0000 README.md edited online with Bitbucket git blame은 개별 파일에서만 작동합니다. 유용한 출력을 위해서는 파일 경로가 필요합니다. git blame의 기본 실행은 명령 도움말 메뉴를 출력합니다. 이 예에서는 README.MD 파일을 사용합니다. 프로젝트의 문서 소스로 git 저장소의 루트에 README 파일을 포함시키는 것은 일반적인 오픈 소스 소프트웨어 관행입니다. git blame README.MD 위의 명령을 실행하면 첫 번째 비난 출력 샘플을 얻을 수 있습니다. 다음 출력은 README의 전체 책임 출력의 하위 집합입니다. 또한이 출력은이 글을 쓰는 시점의 repo 상태를 반영합니다. $ git blame README.md 82496ea3 (kevzettler 2018-02-28 13:37:02 -0800 1) # Git Blame example 82496ea3 (kevzettler 2018-02-28 13:37:02 -0800 2) 89feb84d (Albert So 2018-03-01 00:54:03 +0000 3) This repository is an example of a project with multiple contributors making commits. 82496ea3 (kevzettler 2018-02-28 13:37:02 -0800 4) 82496ea3 (kevzettler 2018-02-28 13:37:02 -0800 5) The repo use used elsewhere to demonstrate `git blame` 82496ea3 (kevzettler 2018-02-28 13:37:02 -0800 6) 89feb84d

2014 - 2023 CURVC Corp. All rights reserved

(Albert So 2018-03-01 00:54:03 +0000 7) Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod TEMPOR incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum 89feb84d (Albert So 2018-03-01 00:54:03 +0000 8) eb06faed (Juni Mukherjee 2018-03-01 19:53:23 +0000 9) Annotates each line in the given file with information from the revision which last modified the line. Optionally, start annotating from the given revision. eb06faed (Juni Mukherjee 2018-03-01 19:53:23 +0000 10) 548dabed (Juni Mukherjee 2018-03-01 19:55:15 +0000 11) Creating a line to support documentation needs for git blame. 548dabed (Juni Mukherjee 2018-03-01 19:55:15 +0000 12) 548dabed (Juni Mukherjee 2018-03-01 19:55:15 +0000 13) Also, it is important to have a few of these commits to clearly reflect the who, the what and the when. This will help Kev get good screenshots when he runs the git blame on this README. 이것은 README.md http://README.md 파일의 처음 13 행의 샘플입니다. 이 출력을 더 잘 이해하기 위해 라인을 구분할 수 있습니다. 다음 표는 3 행의 내용을 표시하며 표의 열은 열의 내용을 나타냅니다. Id Author Timestamp Line Number Line Content 89feb84d Albert So 2018-03-01 00:54:03 +0000 3 This repository is an example of a project with multiple contributors making commits. blame 출력 목록을 검토하면 몇 가지 관찰을 할 수 있습니다. 저자가 세 명 있습니다. 프로젝트의 유지 보수 담당자 인 Kev Zettler 외에도 Albert So, Juni Mukherjee도 나와 있습니다. 저자는 일반적으로 git blame 산출물의 가장 중요한 부분입니다. 타임 스탬프 열은 주로 유용합니다. 변경 내용은 줄 내용 열에 의해 표시됩니다. 일반적인 옵션 git blame -L 1,5 README.md -L 옵션은 출력을 요청 된 행 범위로 제한합니다. 여기에서는 출력을 1-5 행으로 제한했습니다. git blame -e README.md -e 옵션은 사용자 이름 대신 작성자 전자 메일 주소를 표시합니다. git blame -w README.md -w 옵션은 공백 변경을 무시합니다. 이전 작성자가 탭에서 공백으로 바꾸거나 새 줄을 추가하여 파일의 간격을 수정 한 경우 불행히도 이러한 변경 사항을 표시하여 git blame의 결과를가립니다. git blame -M README.md -M 옵션은 동일한 파일 내의 이동 또는 복사 된 행을 감지합니다. 이렇게하면 선을 이동하거나 복사 한 마지막 작성자 대신 선의 원래 작성자가보고됩니다. git blame -C README.md -C 옵션은 다른 파일에서 이동되거나 복사 된 행을 감지합니다. 이렇게하면 선을 이동하거나 복사 한 마지막 작성자 대신 선의 원래 작성자가보고됩니다. Git Blame vs Git Log 634/5000 git blame은 라인을 수정 한 마지막 저자를 표시하지만 종종 원래 라인이 추가 된 시점을 알고 싶을 때가 있습니다. 이것은 git blame을 사용하여 성취하기가 번거로울 수 있습니다. -w, -C 및 -M 옵션을 조합해야합니다. git log 명령을 사용하는 것이 훨씬 더 편리 할 수 있습니다. 특정 코드 조각이 추가되거나 수정 된 모든 원래 커밋을 나열하려면 -S 옵션을 사용하여 git log를 실행하십시오. 찾고있는 코드에 -S 옵션을 추가하십시오. 위의 README 출력에있는 행 중 하나를 예로 들어 보겠습니다. README 출력의 12 행에서 "CSS3D 및 WebGL 렌더러"텍스트를 가져와 봅시다. $ git log -S"CSS3D and WebGL renderers." --pretty=format:'%h %an %ad %s' e339d3c85 Mario Schuettel Tue Oct 13 16:51:06 2015 +0200 reverted README.md to original content 509c2cc35 Daniel Tue Sep 8 13:56:14 2015 +0200 Updated README cb20237cc Mr.doob Mon Dec 31 00:22:36 2012 +0100 Removed DOMRenderer. Now with the CSS3DRenderer it has become irrelevant. 이 결과는 README의 내용이 3 명의 다른 저자에 의해 3 번 추가되거나 수정되었음을 보여줍니다. 그것은 원래 Mr.doob이 커밋 cb20237cc에 추가했습니다. 이 예제에서 git log 앞에 --pretty-format 옵션이 추가되었습니다. 이 옵션은 git log의 기본 출력 형식을 git log의 형식과 일치하는 것으로 변환합니다. 사용법과 설정 옵션에 대한 더 자세한 정보는 git log 페이지를 방문하십시오. Git 가이드

2019-06-16 06:31:03.307

Git Tag

2014 - 2023 CURVC Corp. All rights reserved

tag.png 이 문서는 Git의 태그 지정과 git tag 명령에 대해 설명합니다. 태그는 Git 히스토리의 특정 지점을 가리키는 참조입니다. 태그 지정은 일반적으로 마크 업 된 버전 릴리스 (예 : v1.0.1)에 사용되는 기록의 한 지점을 캡처하는 데 사용됩니다. 태그는 변경되지 않는 분기와 같습니다. 브랜치와는 달리 태그는 생성 된 후에 더 이상 커밋 된 이력이 없습니다. 브랜치에 대한 자세한 정보는 git 브랜치 페이지를 참조하십시오. 이 문서에서는 다양한 종류의 태그, 태그 생성 방법, 모든 태그 나열 방법, 태그 삭제 방법, 태그 공유 방법 등에 대해 다룹니다. 태그 만들기 다음 명령어를 이용해 태그를 만들 수 있습니다. git tag <tagname> <tagname>을 태그가 생성 될 때의 레포 상태에 대한 의미 적 식별자로 바꿉니다. 일반적인 패턴은 git tag v1.4와 같은 버전 번호를 사용하는 것입니다. Git은 두 가지 유형의 태그, 주석이 달린 태그 및 간단한 태그를 지원합니다. 이전 예제에서는 간단한 태그를 만들었습니다. 경량 태그와 주석 태그는 저장된 메타 데이터 양이 다릅니다. 주석이 달린 태그는 공개로, 경량 태그는 비공개로 간주하는 것이 가장 좋습니다. 주석 태그는 태그 이름, 전자 메일 및 날짜와 같은 추가 메타 데이터를 저장합니다. 이것은 공개 자료의 중요한 데이터입니다. 경량 태그는 본질적으로 커밋에 '북마크'이며, 커밋에 대한 포인터와 포인터이며, 관련 커밋에 대한 빠른 링크를 만드는 데 유용합니다. Annotated Tags 주석이 달린 태그는 Git 데이터베이스에 완전한 객체로 저장됩니다. 다시 말하면, 그들은 tagger 이름, 이메일 및 날짜와 같은 추가 메타 데이터를 저장합니다. 커밋 및 커밋 메시지와 유사 주석이 달린 태그에는 태그 지정 메시지가 있습니다. 또한 보안을 위해 주석이 달린 태그는 GNU Privacy Guard (GPG)를 사용하여 서명하고 확인할 수 있습니다. git 태그 지정에 권장되는 권장 사항은 모든 관련 메타 데이터를 가질 수 있도록 가벼운 태그보다 주석 태그를 선호하는 것입니다. git tag -a v1.4 -m "my version 1.4" 이 명령을 실행하면 이전 호출과 유사하지만이 명령 버전에는 -m 옵션과 메시지가 전달됩니다. 이는 git commit -m과 비슷한 편리한 메소드로, 즉시 새 태그를 작성하고 로컬 텍스트 편집기를 열어 -m 옵션과 함께 전달 된 메시지를 저장하지 않아도됩니다. Lightweight Tags git tag v1.4-lw 이 명령을 실행하면 v1.4-lw로 식별되는 경량의 태그가 작성됩니다. 경량 태그는 -a, -s 또는 -m 옵션이 없으면 작성됩니다. 경량 태그는 새 태그 체크섬을 만들어 프로젝트의 repo .git / 디렉토리에 저장합니다. 태그 목록 보기 다음 명령을 통해 저장소에 저장된 태그 목록을 볼 수 있습니다. git tag 명령의 결과는 다음과 같습니다. v0.10.0 v0.10.0-rc1 v0.11.0 v0.11.0-rc1 v0.11.1 v0.11.2 v0.12.0 v0.12.0-rc1 v0.12.1 v0.12.2 v0.13.0 v0.13.0-rc1 v0.13.0-rc2 태그 목록을 구체화하기 위해 와일드 카드 표현식과 함께 -l 옵션을 전달할 수 있습니다. $ git tag -l *-rc* v0.10.0-rc1 v0.11.0-rc1 v0.12.0-rc1 v0.13.0-rc1 v0.13.0-rc2 v0.14.0-rc1 v0.9.0-rc1 v15.0.0-rc.1 v15.0.0-rc.2 v15.4.0-rc.3 이 이전 예제는 -l 옵션과 -rc 접두어로 표시된 모든 태그 목록을 반환하는 -rc의 와일드 카드 표현식을 사용합니다.이 태그는 전통적으로 릴리스 후보를 식별하는 데 사용됩니다. 과거 커밋에 태그 생성하기 이전 태깅 예제는 암시 적 커밋에 대한 작업을 보여주었습니다. 기본적으로 git 태그는 HEAD가 참조하는 커밋에 태그를 생성합니다. 또는 git 태그를 특정 커밋에 대한 참조로 전달할 수 있습니다. HEAD를 기본값으로하는 대신 커밋 된 커밋에 태그를 붙입니다. 이전 커밋 목록을 수집하려면 git log 명령을 실행하십시오. $ git log --pretty=oneline 15027957951b64cf874c3557a0f3547bd83b3ff6 Merge branch 'feature' a6b4c97498bd301d84096da251c98a07c7723e65 add update method for thing 0d52aaab4479697da7686c15f77a3d64d9165190 one more thing 6d52a271eda8725415634dd79daabbc4d9b6008e Merge branch 'experiment' git log를 실행하면 커밋 목록이 출력됩니다. 이 예제에서 우리는 가장 새로운 커밋 Merge branch 'feature'를 선택합니다. Git에 전달하기 위해 SHA 해시 커밋을 참조해야합니다. git tag -a v1.2 15027957951b64cf874c3557a0f3547bd83b3ff6 위의 git 태그 호출을 실행하면 이전의 git 로그 예제에서 선택한 커밋에 대해 v1.2로 식별 된 새로운 주석 된 커밋이 만들어집니다. 태그 다시 하기 / 생성된 태그 교체하기 기존 태그와 동일한 식별자를 가진 태그를 만들려고 시도하면 Git은 다음과 같은 오류를 발생시킵니다. fatal: tag 'v0.4' already exists 또한 이전 커밋에 기존 태그 식별자를 사용하여 태그를 지정하려고하면 Git에서 동일한 오류가 발생합니다. 기존 태그를 갱신해야하는 경우 -f FORCE 옵션을 사용해야합니다. git tag -a -f v1.4 15027957951b64cf874c3557a0f3547bd83b3ff6 위의 명령을 실행하면 15027957951b64cf874c3557a0f3547bd83b3ff6 커밋을 v1.4 태그 식별자에 매핑합니다. v1.4 태그의 기존 내용을 덮어 씁니다. 태그 공유하기: Pushing Tags to Remote 태그를 공유하는 것은 분기를 푸시하는 것과 유사합니다. 기본적으로 git push는 태그를 푸시하지 않습니다. 태그는 git push에 명시 적으로 전달되어야합니다. $ git push origin v1.4 Counting objects: 14, done. Delta compression using up to 8 threads. Compressing objects: 100% (12/12), done. Writing objects: 100% (14/14), 2.05 KiB | 0 bytes/s, done. Total 14 (delta 3), reused 0 (delta 0) To git@bitbucket.com:atlasbro/gittagdocs.git * [new tag] v1.4 -> v1.4 여러 개의 태그를 동시에 푸시하려면 --tags 옵션을 git push 명령에 전달하십시오. 다른 사용자가 Repo를 복제하거나 가져 오면 새 태그가 수신됩니다. 태그 체크아웃하기 git checkout 명령을 사용하여 태그에서 repo의 상태를 볼 수 있습니다. git checkout v1.4 위의 명령은 v1.4 태그를 체크 아웃합니다. 이렇게하면 repo가 분리 된 HEAD 상태가됩니다. 이는 변경 한 내용이 태그를 업데이트하지 않음을 의미합니다. 그들은 새로운 분리 된 커밋을 만들 것입니다. 이 새로운 분리 된 커밋은 모든 분기의 일부가 아니며 SHA 해시 커밋에 의해서만 직접 접근 할 수 있습니다. 그러므로 분리 된 HEAD 상태에서 변경을 수행 할 때마다 새 분기를 작성하는 것이 가장 좋습니다. 태그 삭제하기 태그를 삭제하는 것은 간단합니다. git 태그에 -d 옵션과 태그 식별자를 전달하면 식별 된 태그가 삭제됩니다. $ git tag v1 v2 v3 $ git tag -d v1 $ git tag v2 v3 이 예제에서는 git 태그가 실행되어 v1, v2, v3을 나타내는 태그 목록을 표시 한 다음 v1 태그를 삭제하는 git 태그 -d v1이 실행됩니다. Git 가이드

2014 - 2023 CURVC Corp. All rights reserved

2019-06-16 06:35:24.491

Git Status

tag.png git status 명령은 작업 디렉토리 및 스테이징 영역의 상태를 표시합니다. Git이 어떤 변경 내용을 준비했는지, 어떤 변경 사항이 추적되어 있지 않은지, 어떤 파일이 추적 중인지 확인할 수 있습니다. 상태 출력은 커밋 된 프로젝트 기록과 관련된 정보를 표시하지 않습니다. 이를 위해서는 git log를 사용해야합니다. 사용법 git status Staged, unstaged, 그리고 untracked 파일 목록을 보여줍니다. Ignoring Files 추적 할 수없는 파일은 일반적으로 두 가지 범주로 나뉩니다. 그것들은 방금 프로젝트에 추가되었고 아직 커밋되지 않은 파일이거나 .pyc, .obj, .exe 등의 컴파일 된 바이너리입니다. 이전에는 git에 포함하는 것이 유익했지만 상태 출력, 후자는 실제로 당신의 저장소에서 진행되고있는 것을보기 어렵게 만들 수 있습니다. 이러한 이유로 Git은 .gitignore라는 특수 파일에 경로를 넣어 파일을 완전히 무시할 수 있습니다. 무시하려는 파일은 별도의 줄에 포함해야하며 * 기호는 와일드 카드로 사용할 수 있습니다. 예를 들어 프로젝트 루트의 .gitignore 파일에 다음을 추가하면 컴파일 된 파이썬 모듈이 git 상태로 나타나지 않습니다. Example 뜻하지 않은 것을 실수로 저지르지 않도록 변경하기 전에 저장소의 상태를 확인하는 것이 좋습니다. 이 예에서는 스냅 샷을 준비하고 커밋하기 전과 후에 리포지토리 상태를 표시합니다. # Edit hello.py git status # hello.py is listed under "Changes not staged for commit" git add hello.py git status # hello.py is listed under "Changes to be committed" git commit git status # nothing to commit (working directory clean) 첫 번째 상태 출력은 파일을 unstaged로 표시합니다. 자식 추가 작업은 두 번째 자식 상태에 반영되고 마지막 상태 출력은 커밋 할 것이 없다는 것을 알려주며 작업 디렉토리는 가장 최근의 커밋과 일치합니다. 일부 git 명령 (예 : git merge)은 우연히 변경 사항을 덮어 쓰지 않도록 작업 디렉토리가 깨끗해야합니다. Git 가이드

2021-02-16 16:30:48.602

Git Revert

2014 - 2023 CURVC Corp. All rights reserved

revert.png git revert 명령은 '실행 취소'유형 명령으로 간주 될 수 있지만 기존 실행 취소 작업은 아닙니다. 프로젝트 히스토리에서 커밋을 제거하는 대신 커밋에 의해 도입 된 변경 사항을 반전하는 방법을 알아 내고 역순으로 새로운 커밋을 추가합니다. 이것은 Git이 히스토리를 잃어 버리는 것을 방지하며 히스토리의 무결성과 신뢰성있는 공동 작업에 중요합니다. 되돌리기는 프로젝트 내역에서 커밋의 반대를 적용하고자 할 때 사용해야합니다. 예를 들어 버그를 추적하여 단일 커밋으로 도입 된 경우에 유용합니다. 수동으로 들어가서 수정하고 새 스냅 샷을 커밋하는 대신 git 되돌리기를 사용하여 자동으로이 모든 작업을 수행 할 수 있습니다. 동작 방법 git revert 명령은 저장소의 커밋 기록에 대한 변경 사항을 실행 취소하는 데 사용됩니다. git checkout 및 git reset과 같은 다른 '실행 취소'명령은 HEAD를 이동하고 지정된 포인터로 분기 포인터를 분기합니다. Git revert는 또한 지정된 커밋을 취하지 만, git revert는이 커밋에 대한 ref 포인터를 이동시키지 않습니다. 되돌리기 작업은 지정된 커밋을 취하여 해당 커밋에서 변경 사항을 역으로 적용하고 새로운 "되돌리기 커밋"을 만듭니다. 그런 다음 ref 포인터는 새로운 복귀 커밋을 가리 키도록 업데이트되어 지점의 끝으로 지정됩니다. $ mkdir git_revert_test $ cd git_revert_test/ $ git init . Initialized empty Git repository in /git_revert_test/.git/ $ touch demo_file $ git add demo_file $ git commit -am"initial commit" [master (root-commit) 299b15f] initial commit 1 file changed, 0 insertions(+), 0 deletions(-) create mode 100644 demo_file $ echo "initial content" >> demo_file $ git commit -am"add new content to demo file" [master 3602d88] add new content to demo file n 1 file changed, 1 insertion(+) $ echo "prepended line content" >> demo_file $ git commit -am"prepend content to demo file" [master 86bb32e] prepend content to demo file 1 file changed, 1 insertion(+) $ git log --oneline 86bb32e prepend content to demo file 3602d88 add new content to demo file 299b15f initial commit 여기서는 git_revert_test라는 새로 생성 된 디렉토리에서 repo를 초기화했습니다. demo_file 파일을 추가하고 내용을 두 번 수정 한 repo에 대해 3 번 커밋했습니다. repo 설정 절차가 끝나면 git log를 호출하여 커밋 기록을 표시하고 총 3 개의 커밋을 표시합니다. 이 상태의 repo를 사용하여 git 되돌리기를 시작할 준비가되었습니다. $ git revert HEAD [master b9cd081] Revert "prepend content to demo file" 1 file changed, 1 deletion(-) Git Revert는 커밋 참조가 전달되고 전달되지 않고 실행되지 않을 것으로 기대합니다. 여기서 우리는 HEAD ref를 통과했다. 이렇게하면 최신 커밋을 되돌릴 수 있습니다. 이는 3602d8815dbfa78cd37cd4d189552764b5e96c58을 실행하기 위해 되돌릴 때와 동일한 동작입니다. 병합과 마찬가지로 되돌리기는 새 커밋 메시지를 요구하는 구성된 시스템 편집기를 열 수있는 새로운 커밋을 만듭니다. 커밋 메시지가 입력되고 저장되면 힘내 기능이 작동을 재개합니다. 이제 git log를 사용하여 repo의 상태를 검사하고 이전 로그에 새로운 커밋이 추가되었음을 확인할 수 있습니다. $ git log --oneline 1061e79 Revert "prepend content to demo file" 86bb32e prepend content to demo file 3602d88 add new content to demo file 299b15f initial commit 세 번째 커밋은 되돌리기 후에도 프로젝트 기록에 남아 있습니다. 삭제하는 대신 변경 사항을 취소하기위한 새로운 커밋이 추가되었습니다. 결과적으로 두 번째와 네 번째 커밋은 정확히 같은 코드 기반을 나타내며 세 번째 커밋은 우리가 길을 따라 다시 돌아가고 싶은 경우에 대비하여 우리 역사에 남아 있습니다. 일반적인 옵션 -e --edit 이 옵션은 기본 옵션이므로 지정하지 않아도됩니다. 이 옵션은 구성된 시스템 편집기를 열고 되돌리기를 커밋하기 전에 커밋 메시지를 편집하라는 메시지를 표시합니다. --no-edit 이것은 -e 옵션의 반대입니다. 되돌리기는 편집기를 열지 않습니다. -n --no-commit 이 옵션을 건네면 git revert가 타겟 커밋을 반대로하는 새로운 커밋을 만드는 것을 막을 수 있습니다. 새 커밋을 만드는 대신이 옵션을 사용하면 준비 인덱스 및 작업 디렉터리에 역 변경 사항이 추가됩니다. 이것은 Git이 저장소의 상태를 관리하는 데 사용하는 다른 트리입니다. 자세한 정보는 git reset 페이지를 참조하십시오. Resetting vs. reverting git revert가 단일 커밋을 취소한다는 것을 이해하는 것이 중요합니다. 모든 커밋을 제거함으로써 프로젝트의 이전 상태로 "되돌아 가지"않습니다. 힘내 (Git)에서는 이것을 실제로 리셋 (reset)이라고하며 되돌리기 (revert)라고 부르지 않습니다. revert-1.png 되돌리기에는 재설정보다 두 가지 중요한 이점이 있습니다. 첫째, 공유 저장소에 이미 게시 된 커밋에 대해 "안전한"작업을 수행하는 프로젝트 기록을 변경하지 않습니다. 공유 기록을 변경하는 것이 위험한 이유에 대한 자세한 내용은 [git reset] 페이지를 참조하십시오. 둘째, git revert는 히스토리의 임의의 지점에서 개별 커밋을 대상으로 할 수 있지만 git reset은 현재 커밋에서 뒤로 만 작동 할 수 있습니다. 예를 들어 git reset으로 이전 커밋을 실행 취소하려면 대상 커밋 이후에 발생한 모든 커밋을 제거하고 제거한 다음 모든 커밋을 다시 커밋해야합니다. 말할 필요도없이, 이것은 우아한 실행 취소 솔루션이 아닙니다. git 되돌리기와 다른 '실행 취소'명령의 차이점에 대한 자세한 설명은 재설정, 체크 아웃 및 되돌리기를 참조하십시오. Git 가이드

2019-06-16 12:04:58.79

Git Rm

2014 - 2023 CURVC Corp. All rights reserved

Git을 시작할 때 가장 많이하는 질문은 "Git이 파일을 더 이상 추적하지 않게하려면 어떻게해야합니까?" git rm 명령은 Git 저장소에서 파일을 제거하는 데 사용됩니다. 그것은 git add 명령의 역으로 생각할 수 있습니다. Overview git rm 명령은 개별 파일 또는 파일 모음을 제거하는 데 사용할 수 있습니다. git rm의 주요 기능은 Git 색인에서 추적 된 파일을 제거하는 것입니다. 또한 git rm을 사용하여 스테이징 색인과 작업 디렉토리에서 파일을 제거 할 수 있습니다. 작업 디렉토리에서만 파일을 제거하는 옵션은 없습니다. 조작중인 파일은 현재 HEAD의 파일과 동일해야합니다. 파일의 HEAD 버전과 스테이징 색인 또는 작업 트리 버전간에 불일치가 있으면 Git이 제거를 차단합니다. 이 블록은 진행중인 변경 사항을 제거하지 못하도록하는 안전 메커니즘입니다. git rm은 브랜치를 제거하지 않는다. 사용법 ··· 제거 할 대상 파일을 지정합니다. 옵션 값은 개별 파일, 파일로 구분 된 file1 file2 file3 또는 와일드 카드 파일 glob (~. / directory / *) 일 수 있습니다. -f --force -f 옵션은 HEAD의 파일이 스테이징 색인 및 작업 디렉토리의 현재 내용과 일치하도록하기 위해 힘내 기가 만드는 안전성 검사를 무시하는 데 사용됩니다. -n --dry-run "dry run"옵션은 git rm 명령을 실행하지만 실제로 파일을 삭제하지 않는 보호 장치입니다. 대신 제거 된 파일을 출력합니다. -r -r 옵션은 'recursive'의 줄임말입니다. 재귀 적 모드에서 작동 할 때 rm은 대상 디렉토리와 그 디렉토리의 모든 내용을 제거합니다. -- separator 옵션은 파일 이름 목록과 git rm으로 전달되는 인수를 명시 적으로 구별하는 데 사용됩니다. 일부 파일 이름에 다른 옵션과 혼동 될 수있는 구문이있는 경우 유용합니다. --cached 캐시 옵션은 스테이징 색인에서만 제거가 수행되도록 지정합니다. 작업 디렉토리 파일은 그대로 남습니다. --ignore-unmatch 이로 인해 파일이 일치하지 않아도 명령이 0 sigterm 상태로 종료됩니다. 이것은 유닉스 수준의 상태 코드입니다. 코드 0은 명령의 성공적인 호출을 나타냅니다. --ignore-unmatch 옵션은 정상적으로 실패해야하는 더 큰 쉘 스크립트의 일부로 git rm을 사용할 때 도움이 될 수 있습니다. -q --quiet quiet 옵션은 git rm 명령의 출력을 숨 깁니다. 이 명령은 일반적으로 제거 된 각 파일에 대해 한 줄씩 출력합니다. git rm 취소하기 git rm을 실행하는 것은 영구적 인 업데이트가 아닙니다. 이 명령은 스테이징 색인과 작업 디렉토리를 업데이트합니다. 이러한 변경 사항은 새 확약이 작성되고 변경 사항이 확약 내역에 추가 될 때까지 유지되지 않습니다. 즉, 여기있는 변경 사항은 일반적인 Git 명령을 사용하여 "실행 취소"할 수 있음을 의미합니다. git reset HEAD 리셋하면 현재 스테이징 색인과 작업 디렉토리가 HEAD 커밋으로 되돌아갑니다. 이렇게하면 git rm이 실행 취소됩니다. git checkout . 체크 아웃은 동일한 효과를 가지며 HEAD에서 최신 버전의 파일을 복원합니다. git rm이 실행되었고 제거를 지속하는 새로운 커밋이 생성 된 경우 git reflog를 사용하여 git rm 실행 전의 참조를 찾을 수 있습니다. git reflog 사용에 대해 자세히 알아보십시오. git rm 의 영향 git rm 명령은 현재 분기에서만 작동합니다. 제거 이벤트는 작업 디렉토리 W 스테이징 색인 트리에만 적용됩니다. 파일 h 제는 새 확약이 작성 될 때까지 저장소 실행 기록에 보존되지 않습니다. rm 대신 git rm 을 사용해야하는 이유 Git 저장소는 추적중인 파일에서 일반 셸 rm 명령이 실행된 시점을 인식합니다. 제거를 반영하도록 작업 디렉토리를 업데이트합니다. 제거로 스테이징 색인을 업데이트하지 않습니다. 스테이징 색인에 변경 사항을 추가하려면 제거 된 파일 경로에서 추가 git add 명령을 실행해야합니다. git rm 명령은 작업 디렉토리와 스테이징 색인을 제거와 함께 업데이트한다는 점에서 바로 가기를 수행합니다. 예제 git rm Documentation/₩*.txt 이 예에서는 와일드 카드 파일 glob을 사용하여 Documentation 디렉토리의 하위 디렉토리 및 해당 하위 디렉토리의 모든 *.txt 파일을 제거합니다. 이 예에서는 별표 (*)가 슬래시로 이스케이프됩니다. 이것은 셸이 와일드 카드를 확장하지 못하게하는 가드입니다. 그런 다음 와일드 카드는 Documentation / 디렉토리 아래에있는 파일 및 하위 디렉토리의 경로 이름을 확장합니다. git rm -f git-*.sh 더이상 파일시스템에 존재하지 않는 파일 제거하기 "rm 대신 git rm을 사용하는 이유"에서 설명한 것처럼 실제로 git rm은 표준 쉘 rm과 git add를 결합하여 작업 디렉토리에서 파일을 제거하고 해당 제거를 스테이징 색인으로 승격시키는 편의 명령입니다. 표준 쉘 rm 명령 만 사용하여 여러 파일이 제거 된 경우 저장소가 성가신 상태가 될 수 있습니다. 의도적으로 제거 된 모든 파일을 다음 커밋의 일부로 기록하려는 경우 git commit -a는 다음 커밋 준비를 위해 모든 제거 이벤트를 스테이징 인덱스에 추가합니다. 그러나 쉘 rm을 사용하여 제거 된 파일을 지속적으로 제거하려는 경우 다음 명령을 사용하십시오. git diff --name-only --diff-filter=D -z | xargs -0 git rm --cached 이 명령은 작업 디렉토리에서 제거 된 파일 목록을 생성하고 그 목록을 git rm --cached에 저장하여 스테이징 색인을 업데이트합니다. Git 가이드

2023-08-11 10:14:01.367

Git Clean

clean.png 이 절에서는 git clean 명령에 대해 자세히 설명합니다. 이러한 다른 명령은 이전에 Git 추적 색인에 추가 된 파일에서 작동하지만 git clean 명령은 추적되지 않은 파일에서 작동합니다. 추적 할 수없는 파일은 repo의 작업 디렉토리에서 작성되었지만 아직 git add 명령을 사용하여 저장소의 추적 색인에 추가되지 않은 파일입니다. 추적 된 파일과 추적되지 않은 파일의 차이를보다 잘 보여주기 위해 다음 명령 줄 예제를 참고 하세요. $ mkdir git_clean_test $ cd git_clean_test/ $ git init . Initialized empty Git repository in /Users/kev/code/git_clean_test/.git/ $ echo "tracked" > ./tracked_file $ git add ./tracked_file $ echo "untracked" > ./untracked_file $ mkdir ./untracked_dir && touch ./untracked_dir/file $ git status On branch master Initial commit Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: tracked_file Untracked files: (use "git add <file>..." to include in what will be committed) untracked_dir/

2014 - 2023 CURVC Corp. All rights reserved

untracked_file 이 예제는 git_clean_test 디렉토리에 새로운 Git 저장소를 생성합니다. 그런 다음, Git 색인에 추가 된 tracked_file을 작성하고, untracked_file이 작성되고 untracked_dir이 작성됩니다. 그런 다음 git status를 호출하여 Git의 추적 상태 및 추적되지 않은 변경 상태를 나타내는 출력을 표시합니다. 이 상태의 저장소를 사용하여 git clean 명령을 실행하여 의도 된 목적을 입증 할 수 있습니다. $ git clean fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to clean 이 시점에서 기본 git clean 명령을 실행하면 치명적인 오류가 발생할 수 있습니다. 위의 예는 이것이 어떻게 생겼는지를 보여줍니다. 기본적으로 Git은 시작하기 위해 git clean에 "force"옵션이 전달되도록 전역 적으로 구성됩니다. 이는 중요한 안전 장치입니다. 마지막으로 git clean을 실행하면 실행 취소 할 수 없습니다. 완전히 실행되면 git clean은 명령 줄 rm 유틸리티를 실행하는 것과 마찬가지로 하드 파일 시스템을 삭제합니다. untracked 파일을 실행하기 전에 정말로 삭제하고 싶은지 확인하십시오. 일반적인 옵션과 사용법 기본 git 정리 동작 및 경고에 대한 이전 설명이 주어지면 다음 내용은 다양한 git clean 사용 사례와 해당 작업에 필요한 명령 줄 옵션을 보여줍니다. -n -n 옵션은 git clean의 "dry run"을 수행합니다. 이렇게하면 실제로 파일을 제거하지 않고 어떤 파일을 제거 할 것인지 표시됩니다. 항상 제일 먼저 git clean을 수행하는 것이 가장 좋습니다. 이전에 만든 데모 저장소에서이 옵션을 시연 할 수 있습니다. $ git clean -n Would remove untracked_file git clean 명령을 실행하면 untracked_file이 제거됩니다. untracked_dir은 여기서 출력되지 않습니다. 기본적으로 git clean은 디렉토리에서 재귀 적으로 작동하지 않습니다. 우발적 인 영구 삭제를 방지하는 또 다른 안전 메커니즘입니다. -f or --force force 옵션은 현재 디렉토리에서 untracked 파일의 실제 삭제를 시작합니다. clean.requireForce 구성 옵션을 false로 설정하지 않으면 force가 필요합니다. 이렇게하면 .gitignore에서 지정한 추적되지 않은 폴더 나 파일이 제거되지 않습니다. 이제 우리의 예제 저장소에서 깨끗한 git clean을 실행 해 보겠습니다. $ git clean -f Removing untracked_file 명령은 제거 된 파일을 출력합니다. untracked_file이 제거 된 것을 볼 수 있습니다. 이 시점에서 git status를 실행하거나 ls를 실행하면 untracked_file이 삭제되었고 아무 것도 발견되지 않는다는 것을 알 수 있습니다. 기본적으로 git clean -f는 현재 디렉토리가 아닌 모든 파일에 대해 작동합니다. 또한 특정 파일을 제거하는 -f 옵션과 함께 <path> 값을 전달할 수 있습니다. git clean -f <path> -d include directories -d 옵션은 git clean에게 추적되지 않은 디렉토리를 지우고 싶다고 알려주며 기본적으로 디렉토리를 무시합니다. 앞의 예제에 -d 옵션을 추가 할 수 있습니다. $ git clean -dn Would remove untracked_dir/ $ git clean -df Removing untracked_dir/ 여기서 우리는 untracked_dir을 출력하는 -dn 조합을 사용하여 'dry run'을 실행했습니다. 그런 다음 강제 정리를 실행하고 untracked_dir이 제거 된 출력을 수신합니다. -x force removal of ignored files 일반적인 소프트웨어 릴리스 패턴은 리포지토리 추적 색인에 커밋되지 않은 빌드 또는 배포 디렉토리를 갖는 것입니다. 빌드 디렉토리에는 커밋 된 소스 코드에서 생성 된 임시 빌드 아티팩트가 포함됩니다. 이 빌드 디렉토리는 대개 저장소 .gitignore 파일에 추가됩니다. 추적되지 않은 다른 파일로도이 디렉토리를 정리하는 것이 편리 할 수 있습니다. -x 옵션은 git clean에 무시 된 파일을 포함하도록 지시합니다. 이전의 git clean 호출과 마찬가지로, 마지막 삭제 전에 먼저 'dry run'을 실행하는 것이 가장 좋습니다. -x 옵션은 프로젝트 빌드와 관련된 모든 파일에 적용됩니다. 이것은 ./.idea IDE 구성 파일과 같은 의도하지 않은 것일 수 있습니다. git clean -xf -d 옵션과 마찬가지로 -x는 다른 옵션과 함께 전달되고 작성 될 수 있습니다. 이 예제는 -f 옵션과 함께 현재 디렉토리에서 추적되지 않은 파일과 Git이 일반적으로 무시하는 파일을 제거하는 방법을 보여줍니다. Interactive mode or git clean interactive 지금까지 살펴본 ad-hoc 명령 줄 실행 외에도 git clean에는 -i 옵션을 전달하여 시작할 수있는 "대화 형"모드가 있습니다. 이 문서가 소개 된 예제 repo를 다시 살펴 보겠습니다. 초기 상태에서는 대화식 클린 세션을 시작합니다. $ git clean -di Would remove the following items: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> -d 옵션을 사용하여 대화식 세션을 시작 했으므로 untracked_dir에도 적용됩니다. 대화식 모드는 명령을 추적 할 수없는 파일에 적용 할 것을 요구하는 What now> 프롬프트를 표시합니다. 명령 자체는 상당히 자명합니다. 우리는 명령 6 : help로 시작하여 각각을 임의의 순서로 간략하게 살펴볼 것입니다. 명령 6을 선택하면 다른 명령이 더 설명됩니다. What now> 6 clean - start cleaning filter by pattern - exclude items from deletion select by numbers - select items to be deleted by numbers ask each - confirm each deletion (like "rm -i") quit - stop cleaning help - this screen ? - help for prompt selection 5: quit 똑바로 앞으로 대화 형 세션을 종료합니다. 1: clean 표시된 항목을 삭제합니다. 이 시점에서 1 : clean을 실행하면 untracked_dir / untracked_file이 제거됩니다. 4: ask each untracked 각 파일을 반복하고 삭제를 묻는 Y / N 프롬프트를 표시합니다. 다음과 같이 보입니다. *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 4 Remove untracked_dir/ [y/N]? N Remove untracked_file [y/N]? N 2: filter by pattern Will display an additional prompt that takes input used to filter the list of untracked files. Would remove the following items: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 2 untracked_dir/ untracked_file Input ignore patterns>> *_file untracked_dir/ 여기서 우리는 * _file 와일드 카드 패턴을 입력하여 untracked 파일 목록을 untracked_dir로 제한합니다. 3: select by numbers 명령 2와 마찬가지로 명령 3은 추적되지 않은 파일 이름의 목록을 수정합니다. 대화식 세션에서는 추적 할 수없는 파일 이름에 해당하는 번호를 묻습니다. Would remove the following items: untracked_dir/ untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> 3 1: untracked_dir/ 2: untracked_file Select items to delete>> 2 1: untracked_dir/ * 2: untracked_file Select items to delete>> Would remove the following item: untracked_file *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help Git 가이드

2014 - 2023 CURVC Corp. All rights reserved

2020-04-22 09:06:47.606

Git 실행 취소

undo.png 이 섹션에서는 사용 가능한 '실행 취소'Git 전략과 명령에 대해 설명합니다. Git은 워드 프로세싱 응용 프로그램에서 발견되는 것과 같은 전통적인 '실행 취소'시스템이 없다는 점에 유의해야합니다. Git 작업을 전통적인 '실행 취소'정신 모델에 매핑하는 것을 자제하는 것이 좋습니다. 또한 Git에는 '실행 취소'작업에 대한 고유 한 명명 규칙이 있으므로 이를 활용하는 것이 가장 좋습니다. 이 용어에는 재설정, 되돌리기, 체크 아웃, 정리 등과 같은 용어가 포함됩니다. 재미있는 은유는 Git을 타임 라인 관리 유틸리티로 생각하는 것입니다. 커밋은 프로젝트 히스토리의 타임 라인을 따라 특정 시점 또는 관심 지점의 스냅 샷입니다. 또한 분기를 사용하여 여러 타임 라인을 관리 할 수 있습니다. Git에서 '실행 취소'할 때, 보통 시간 내에 다시 이동하거나 실수가 발생하지 않은 다른 타임 라인으로 이동합니다. 이 튜토리얼에서는 소프트웨어 프로젝트의 이전 버전을 사용하기 위해 필요한 모든 기술을 제공합니다. 첫째, 이전 커밋을 탐색하는 방법을 보여주는 다음 프로젝트 기록에서 공용 커밋을 되 돌리는 것과 로컬 컴퓨터에서 게시되지 않은 변경을 다시 설정하는 것의 차이점을 설명합니다. Finding what is lost: Reviewing old commits 모든 버전 제어 시스템의 기본 아이디어는 프로젝트의 "안전한"복사본을 저장하여 코드 기반을 돌이킬 수 없을 정도로 파괴 할 필요가 없다는 것입니다. 커밋의 프로젝트 히스토리를 구축하면 히스토리에서 커밋을 검토하고 다시 방문 할 수 있습니다. Git 저장소의 히스토리를 검토하는 데 가장 좋은 유틸리티 중 하나는 git log 명령입니다. 아래 예제에서 git log를 사용하여 널리 사용되는 오픈 소스 그래픽 라이브러리에 대한 최신 커밋 목록을 가져옵니다. git log --oneline e2f9a78fe Replaced FlyControls with OrbitControls d35ce0178 Editor: Shortcuts panel Safari support. 9dbe8d0cf Editor: Sidebar.Controls to Sidebar.Settings.Shortcuts. Clean up. 05c5288fc Merge pull request #12612 from TyLindberg/editor-controls-panel 0d8b6e74b Merge pull request #12805 from harto/patch-1 23b20c22e Merge pull request #12801 from gam0022/improve-raymarching-example-v2 fe78029f1 Fix typo in documentation 7ce43c448 Merge pull request #12794 from WestLangley/dev-x 17452bb93 Merge pull request #12778 from OndrejSpanel/unitTestFixes b5c1b5c70 Merge pull request #12799 from dhritzkiv/patch-21 1b48ff4d2 Updated builds. 88adbcdf6 WebVRManager: Clean up. 2720fbb08 Merge pull request #12803 from dmarcos/parentPoseObject 9ed629301 Check parent of poseObject instead of camera 219f3eb13 Update GLTFLoader.js 15f13bb3c Update GLTFLoader.js 6d9c22a3b Update uniforms only when onWindowResize 881b25b58 Update ProjectionMatrix on change aspect 각 커밋에는 해시를 식별하는 고유 한 SHA-1이 있습니다. 이러한 ID는 커밋 된 타임 라인을 통해 이동하고 커밋을 다시 방문하는 데 사용됩니다. 기본적으로 git log는 현재 선택된 브랜치에 대해서만 커밋을 보여준다. 찾고자하는 커밋이 다른 지사에 있다는 것은 전적으로 가능합니다. git log --branches = *를 실행하여 모든 브랜치에서 모든 커밋을 볼 수 있습니다. git 브랜치 명령은 다른 브랜치를보고 방문하는 데 사용됩니다. 명령을 호출하면, git branch -a는 모든 알려진 브랜치 이름의 목록을 리턴합니다. git log <branch_name>을 사용하여 이러한 분기 이름 중 하나를 기록 할 수 있습니다. 히스토리에서 방문하고자하는 지점에 대한 커밋 참조를 발견하면 git checkout 명령을 사용하여 커밋을 방문 할 수 있습니다. 힘내 체크 아웃은 저장된 스냅 샷을 개발 컴퓨터에 "로드"하는 쉬운 방법입니다. 일반적인 개발 과정에서 HEAD는 대개 마스터 또는 다른 로컬 지사를 가리 킵니다. 그러나 이전 커밋을 체크 아웃 할 때 HEAD는 더 이상 지사를 가리 키지 않습니다. 바로 커밋을 가리 킵니다. 이를 "분리 된 HEAD"상태라고하며 다음과 같이 시각화 할 수 있습니다. co-1.png 이전 파일을 체크 아웃해도 HEAD 포인터는 이동하지 않습니다. 동일한 분기와 동일한 커밋에 남으며 '분리 된 헤드'상태를 피합니다. 그런 다음 다른 모든 변경 사항과 마찬가지로 새 스냅 샷에서 이전 버전의 파일을 커밋 할 수 있습니다. 결과적으로 파일에서 git checkout을 사용하면 개별 파일의 이전 버전으로 되돌릴 수 있습니다. 이 두 모드에 대한 더 자세한 정보는 git checkout 페이지를 방문하십시오. 과거 커밋 보기 이 예에서는 미친 실험을 시작했다고 가정하지만 계속 유지할지 여부는 확실하지 않습니다. 결정을 돕기 위해 실험을 시작하기 전에 프로젝트의 상태를 살펴보고 싶습니다. 먼저, 보려는 리비전의 ID를 찾아야합니다. git log --oneline 프로젝트 내역이 다음과 같다고 가정 해 보겠습니다. b7119f2 Continue doing crazy things 872fa7e Try something crazy a1e8fb5 Make some important changes to hello.txt 435b61d Create hello.txt 9773e52 Initial import git checkout을 사용하여 다음과 같이 "hello.txt에 대한 일부 변경 사항 가져 오기"커밋을 볼 수 있습니다. git checkout a1e8fb5 이렇게하면 작업 디렉토리가 a1e8fb5 커밋의 정확한 상태와 일치하게됩니다. 프로젝트의 현재 상태를 잃지 않고 걱정없이 파일을보고 프로젝트를 컴파일하고 테스트를 실행하고 파일을 편집 할 수 있습니다. 여기서 수행하는 작업은 저장소에 저장되지 않습니다. 계속 개발하려면 프로젝트의 "현재"상태로 돌아 가야합니다. git checkout master 여기서는 기본 마스터 분기에서 개발중인 것으로 가정합니다. 일단 master 브랜치로 돌아 왔으면 git revert 나 git reset을 사용하여 원하지 않는 변경을 취소 할 수 있습니다. Undoing a committed snapshot 기술적으로 커밋을 '실행 취소'하는 여러 가지 전략이 있습니다. 다음 예제는 우리가 다음과 같은 커밋 기록을 가지고 있다고 가정합니다 : git log --oneline 872fa7e Try something crazy a1e8fb5 Make some important changes to hello.txt 435b61d Create hello.txt 9773e52 Initial import 우리는 872fa7e를 실행 취소하는 데 초점을 맞출 것입니다. 어쩌면 일들이 너무 미쳤을 수도 있습니다. How to undo a commit with git checkout git checkout 명령을 사용하여 이전 커밋 인 a1e8fb5를 체크 아웃 할 수 있습니다. 미친 커밋이 발생하기 전에 저장소에 상태를 저장합니다. 특정 커밋을 체

2014 - 2023 CURVC Corp. All rights reserved

크 아웃하면 repo가 "분리 된 헤드"상태가됩니다. 즉, 어떤 지점에서도 더 이상 일하지 않습니다. 분리 된 상태에서, 분기를 확립 된 분기로 다시 변경할 때 사용자가 작성하는 모든 새로운 확약은 고아가됩니다. Git의 가비지 컬렉터가 고아가 된 커밋을 삭제합니다. 가비지 수집기는 구성된 간격으로 실행되고 고아 커밋을 영구적으로 파괴합니다. 고아 커밋이 가비지 수집되지 않게하려면 우리가 지점에 있는지 확인해야합니다. 분리 된 HEAD 상태에서 git checkout -b new_branch_without_crazy_commit을 실행할 수 있습니다. 그러면 new_branch_without_crazy_commit이라는 새 분기가 만들어지고 그 상태로 전환됩니다. repo는 이제 872fa7e 커밋이 더 이상 존재하지 않는 새로운 히스토리 타임라인에 있습니다. 이 시점에서 우리는 872fa7e 커밋이 더 이상 존재하지 않고 '취소된'것으로 간주하는이 새로운 분기에 대한 작업을 계속할 수 있습니다. 불행히도, 이전 분기가 필요하다면, 아마도 마스터 브랜치 일 것입니다.이 실행 취소 전략은 적절하지 않습니다. 다른 '실행 취소'전략을 살펴 보겠습니다. 더 많은 정보와 예제는 우리의 심도있는 git checkout 토론을 검토한다. How to undo a public (pushed) commit with git revert 우리가 원래의 커밋 히스토리 예제로 돌아가고 있다고 가정합시다. 872fa7e 커밋을 포함하는 기록. 이번에는 되돌리기 '실행 취소'를 시도해 보겠습니다. git revert HEAD를 실행하면, Git은 마지막 커밋의 역함수로 새로운 커밋을 생성합니다. 이렇게하면 현재 분기 기록에 새 커밋이 추가되고 다음과 같이 표시됩니다. git log --oneline e2f9a78 Revert "Try something crazy" 872fa7e Try something crazy a1e8fb5 Make some important changes to hello.txt 435b61d Create hello.txt 9773e52 Initial import 이 시점에서 우리는 다시 기술적으로 '취소 된'872fa7 커밋을 수행했습니다. 역사적으로 872fa7e가 존재하지만, 새로운 e2f9a78 커밋은 872fa7e의 변경과 반대입니다. 이전의 결제 전략과 달리 동일한 지점을 계속 사용할 수 있습니다. 이 솔루션은 만족스러운 실행 취소입니다. 이것은 공용 공유 리포지토리를 사용하기위한 이상적인 '실행 취소'방법입니다. 큐레이터와 최소한의 Git 히스토리를 유지해야하는 경우이 전략은 만족스럽지 않을 수 있습니다. How to undo a commit with git reset 이 실행 취소 전략에 대해서는 작업 예제를 계속 진행합니다. git reset은 여러 용도와 기능을 가진 광범위한 명령입니다. git reset --hard a1e8fb5를 호출하면 커밋 기록이 지정된 커밋으로 재설정됩니다. git log로 커밋 내역을 검사하면 다음과 같이 표시됩니다. git log --oneline a1e8fb5 Make some important changes to hello.txt 435b61d Create hello.txt 9773e52 Initial import 로그 출력은 e2f9a78 및 872fa7e 커밋이 더 이상 커밋 기록에 존재하지 않음을 나타냅니다. 이 시점에서 우리는 작업을 계속하고 마치 "미친"커밋이 발생하지 않은 것처럼 새로운 커밋을 만들 수 있습니다. 변경 사항을 취소하는이 방법은 기록에 가장 명확한 영향을줍니다. 리셋을 수행하는 것은 로컬 변경에 유용하지만 공유 원격 리포지토리로 작업 할 때 문제를 일으 킵니다. 커밋 872fa7e이 푸시된 공유 저장소가 있고, 해당 저장소에 히스토리가 reset 된 브랜치를 git push 하려고 시도하면 Git이 이를 잡아 오류를 던질 것이다. Git은 푸시 된 브랜치가 커밋이 누락되어 최신 상태가 아니라고 가정합니다. 이러한 시나리오에서 git revert 로 실행을 취소해야 합니다. Undoing the last commit 이전 섹션에서는 커밋을 취소하기위한 다양한 전략에 대해 논의했습니다. 이러한 전략은 모두 최신 커밋에도 적용됩니다. 그러나 어떤 경우에는 마지막 커밋을 제거하거나 재설정 할 필요가 없을 수도 있습니다. 어쩌면 그것은 조숙하게 만들어 졌을지도 모릅니다. 이 경우 가장 최근의 커밋을 수정할 수 있습니다. git add를 사용하여 작업 디렉토리에서 더 많은 변경을하고 커밋을 위해 준비하면 git commit --amend를 실행할 수 있습니다. 이것은 Git에 구성된 시스템 편집기를 열고 마지막 커밋 메시지를 수정할 수있게 합니다. 새로운 변경 사항이 수정 된 커밋에 추가됩니다. Undoing uncommitted changes 변경 사항이 저장소 히스토리에 커밋되기 전에 스테이징 색인 및 작업 디렉토리에 살고 있습니다. 이 두 영역에서 변경 사항을 실행 취소해야 할 수도 있습니다. 스테이징 색인과 작업 디렉토리는 내부의 Git 상태 관리 메커니즘입니다. 이 두 메커니즘이 어떻게 작동하는지에 대한 자세한 내용을 보려면 git reset 페이지를 방문하십시오. The working directory 작업 디렉토리는 일반적으로 로컬 파일 시스템과 동기화됩니다. 작업 디렉토리의 변경 사항을 취소하려면 일반적으로 좋아하는 편집기를 사용하는 것처럼 파일을 편집 할 수 있습니다. 힘내는 작업 디렉토리를 관리하는 데 도움이되는 몇 가지 유틸리티를 가지고있다. 작업 디렉토리의 변경을 취소하기위한 편리한 유틸리티 인 git clean 명령이 있습니다. 또한, git reset은 --mixed 또는 --hard 옵션을 사용하여 호출 할 수 있으며 작업 디렉토리에 재설정을 적용합니다. The staging index git add 명령은 스테이징 색인에 변경 사항을 추가하는 데 사용됩니다. Git 재설정은 주로 준비 인덱스 변경 사항을 실행 취소하는 데 사용됩니다. -혼합 리셋은 보류중인 모든 변경 사항을 스테이징 인덱스에서 작업 디렉토리로 다시 이동시킵니다. Undoing public changes 원격 리포지토리가있는 팀에서 작업 할 때는 변경 사항을 취소 할 때 추가 고려해야합니다. Git 재설정은 일반적으로 '로컬'실행 취소 방법으로 간주되어야합니다. 사설 분기에 대한 변경을 취소 할 때 리셋을 사용해야합니다. 이렇게하면 다른 개발자가 사용하고있는 다른 브랜치에서 커밋을 안전하게 제거 할 수 있습니다. 문제는 공유 분기에서 리셋이 실행될 때 발생하며 그 브랜치는 git push를 사용하여 원격으로 푸시됩니다. 이 시나리오의 경우 Git은 푸시 된 브랜치가 커밋을 잃어 버렸기 때문에 원격 브랜치에 비해 오래된 것으로 인식하여 푸시를 차단할 것이다. 공유 기록을 실행 취소하는 기본 방법은 git revert 입니다. 되돌리기는 공유 기록에서 커밋을 제거하지 않기 때문에 재설정보다 안전합니다. 되돌리기는 실행 취소하려는 커밋을 유지하고 원하지 않는 커밋을 반전하는 새로운 커밋을 만듭니다. 이 방법은 공유 된 원격 공동 작업에서 더 안전합니다. 원격 개발자가 분기를 가져 와서 원하지 않는 커밋을 실행 취소하는 새로운 되돌리기 커밋을 수신 할 수 있기 때문입니다. Git 가이드

2019-06-16 11:45:37.879

2014 - 2023 CURVC Corp. All rights reserved

Git Reset

git reset 명령은 변경 사항을 실행 취소하기위한 복잡하고 다양한 도구입니다. 세 가지 기본 형식의 호출이 있습니다. 이러한 형식은 명령 줄 인수 --soft, --mixed, --hard에 해당합니다. 세 개의 인수는 각각 Git의 세 가지 내부 상태 관리 메커니즘 인 HEAD (The Commit Tree), 스테이징 색인 및 작업 디렉토리에 해당합니다. Git reset & three trees of Git git reset 사용법을 제대로 이해하려면 Git의 내부 상태 관리 시스템을 먼저 이해해야합니다. 때로는 이러한 메커니즘을 Git의 "three trees"라고 부릅니다. 나무는 엄격하게 전통적인 트리 데이터 구조가 아니므로 잘못된 이름일 수 있습니다. 그러나 Git은 편집 타임 라인을 추적하는 데 사용하는 노드 및 포인터 기반 데이터 구조입니다. 이러한 메커니즘을 증명하는 가장 좋은 방법은 리포지토리에 변경 집합을 만들어 세 개의 트리를 따라 이동하는 것입니다. Undo a change with git reset 시작하려면 역사에서 가장 최근 커밋을 실행 취소하십시오. 이 경우 Bitbucket의 CI / CD 솔루션 파이프 라인을 활성화했지만 스크립트가 올바르지 않다는 것을 알았습니다. 터미널 창에 git log --oneline을 입력하십시오. 두 번째 커밋의 커밋 해시를 로그에 복사합니다 (52f823c). 그런 다음 q를 눌러 로그를 종료합니다. git reset --soft 52f823c를 터미널 창에 입력하십시오. 명령이 성공하면 명령이 백그라운드에서 실행됩니다. 그게 전부입니다. 당신은 당신의 첫 번째 변화를 취소했습니다. 이제이 작업의 결과를 보도록하겠습니다. 터미널 창에 git status를 입력하면 커밋이 취소되었고 커밋되지 않은 변경 사항이 표시됩니다. 다음과 같이 보일 것입니다 : $ git status On branch master Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded. (use "git pull" to update your local branch) Changes to be committed: (use "git reset HEAD <file>..." to unstage) new file: bitbucket-pipelines.yml Enter git log --oneline in your terminal window. You should see something like this: $ git log --oneline 52f823c repeated quote to show how a change moves through the process 4801b87 Merged in changes (pull request #6) 1a6a403 myquote edited online with Bitbucket 3b29606 (origin/changes) myquote2.html edited online with Bitbucket 8b236d9 myquote edited online with Bitbucket 235b9a7 testing prs c5826da more changes 43a87f4 remivng d5c4c62 a few small changes 23a7476 Merged in new-feature2 (pull request #3) 5cc4e1e add a commit message cbbb5d6 trying a thing 438f956 adding section for permissions and cleaning up some formatting 23251c1 updated snipptes.xml organization into resources. other files misc changes 3f630f8 Adding file to track changes ... 분기의 새로운 HEAD가 사용자가 원하는대로 커밋 52f823c임을 알 수 있습니다. q를 눌러 로그를 종료하십시오. 간단한 재설정을 수행하는 방법을 배웠으니 이제 좀 더 복잡한 것을 시도해보십시오. 터미널을 열어 두십시오. Undo several changes with git reset 당겨 받기 요청 # 6 (4801b87)을 다시 작성해야하고 git reset 명령을 사용할 때 HEAD가 1a6a403을 커밋하도록 재설정해야한다는 사실을 깨닫게되었다고 가정 해 보겠습니다. git log --online을 입력하십시오. 실행 취소하려는 변경 사항이있는 끌어 오기 요청 # 6 바로 아래 커밋 인 커밋 해시 1a6a403 (Bitbucket을 사용하여 온라인으로 편집 된 myquote)을 복사합니다. 터미널 창에 git reset 1a6a403을 입력하십시오. 출력은 다음과 같이 보일 것입니다 : $ git reset 1a6a403 Unstaged changes after reset: M README.md M myquote2.html 변경 사항이 현재 커밋되지 않은 상태에 있음을 알 수 있습니다. 이는 이제 프로젝트 기록과 준비 영역에서 몇 가지 변경 사항을 제거했음을 의미합니다. Enter git status in your terminal window. The output should look something like this: $ git status On branch master Your branch is behind 'origin/master' by 6 commits, and can be fast-forwarded. (use "git pull" to update your local branch) Changes not staged for commit: (use "git add <file>..." to update what will be committed) (use "git checkout -- <file>..." to discard changes in working directory) modified: README.md modified: myquote2.html Untracked files: (use "git add <file>..." to include in what will be committed) bitbucket-pipelines.yml no changes added to commit (use "git add" and/or "git commit -a") 이제 우리가 undid (bitbucket-pipelines.yml 파일)의 첫 번째 변경 사항이 git에서 완전히 추적 할 수 없음을 알 수 있습니다. git reset을 호출하면 브랜치의 헤드와 git의 추적 영역 또는 인덱스 영역 모두에서 변경 사항이 제거되기 때문입니다. 기본 프로세스는 여기에서 다룰 수있는 것보다 조금 복잡합니다. git reset에서 더 많은 것을 읽을 수 있습니다. Enter git log --oneline in your terminal window. 1a6a403 myquote edited online with Bitbucket 8b236d9 myquote edited online with Bitbucket 43a87f4 remivng d5c4c62 a few small changes 23a7476 Merged in new-feature2 (pull request #3) 5cc4e1e add a commit message cbbb5d6 trying a thing 438f956 adding section for permissions and cleaning up some formatting 23251c1 updated snipptes.xml organization into resources. other files misc changes 3f630f8 Adding file to track changes e52470d README.md edited online with Bitbucket e2fad94 README.md edited online with Bitbucket 592f84f Merge branch 'master' into new-feature2 Merge branch especially if it merges an updated upstream into a topic branch. 7d0bab8 added a line 879f965 adding to the quote file 8994332 Merged in HOT-235 (pull request #2) b4a0b43 removed sarcastic remarks because they violate policy. b5f5199 myquote2.html created online with Bitbucket b851618 adding my first file 5b43509 writing and using tests 로그 출력은 커밋 내역이 수정되어 커밋 1a6a403에서 시작됨을 보여줍니다. 데모 및 추가 예제를 위해 방금 수행 한 재설정을 실행 취소하려고한다고 가정 해 봅시다. 추가 고려 사항을 고려한 후에 풀 요청 # 6의 내용을 유지하려고했습니다. Pushing resets to Remote Git reset은 변경 을 취소하는 방법 중 하나입니다. git reset은 일반적으로 변경 사항을 취소하기위한 '안전하지 않은'옵션으로 간주됩니다. 격리 된 코드에서 로컬로 작업 할 때 재설정은 좋지만 팀 구성원과 공유 할 경우 위험 해집니다. 원격 팀으로 리셋 된 지점을 공유하려면 '강제 푸시'가 실행되어야합니다. 'forced push'는 git push -f를 실행하여 시작됩니다. 강제 푸시는 푸시 시점 이후에 만들어진 branch의 모든 기록을 삭제합니다. '안전하지 않은'시나리오의 예는 다음과 같습니다. Dev A는 새로운 기능을 개발하는 브랜치에서 작업하고 있습니다. Dev B는 별도의 기능을 개발하는 동일한 브랜치에서

2014 - 2023 CURVC Corp. All rights reserved

작업하고 있습니다. Dev B는 Dev A와 Dev B가 모두 작동하기 전에 브랜치를 이전 상태로 재설정하기로 결정합니다. Dev B 그러면 강제로 재설정 브랜치를 원격 저장소로 푸시합니다. Dev A는 분기를 가져 와서 업데이트를받습니다. git pull 동안 Dev A는 강제 업데이트를받습니다. 이것은 기능 작업이 완료되고 커밋을 잃기 전에 Dev A의 로컬 브랜치를 원래대로 되돌립니다. Undo a git reset 지금까지 우리는 git commit에 Sha 해쉬를 보내고있다. 이제 git log 출력에 재설정 된 커밋이 없습니다. 어떻게 그 커밋을 되 찾을 수 있습니까? Git은 커밋을 지우지 않습니다. 또한 git은 "reflog"라고 불리는 모든 ref 동작의 개별 로그를 저장합니다. git reflog를 실행하여 reflog를 검사 할 수 있습니다. 1a6a403 HEAD@{0}: reset: moving to 1a6a403 1f08a70 HEAD@{1}: reset: moving to origin/master 1f08a70 HEAD@{2}: clone: from git@bitbucket.org:dans9190/tutorial-documentation-tests.git git reflog의 결과는 위와 비슷해야합니다. 레포에서 수행한 내역을 볼 수 있습니다. 맨 위 줄은 풀 요청 # 6을 재설정하기 위해 수행 한 재설정에 대한 참조입니다. 이제 풀 리퀘스트 # 6을 리셋하기 위해 리셋을 리셋하자. 이 reflog 출력의 두 번째 열은 repo에서 수행하는 수정 작업에 대한 참조 포인터를 나타냅니다. 여기서 HEAD @ {0}은 이전에 실행 한 재설정 명령에 대한 참조입니다. 리셋 명령을 회신하지 않으려면 HEAD @ {1} (으)로 리포를 복원하십시오. $ git reset --hard HEAD@{1} HEAD is now at 1f08a70 Initial Bitbucket Pipelines configuration 이제 git log --oneline을 사용하여 repos commit history를 살펴 보겠습니다: $git log --online 1f08a70 Initial Bitbucket Pipelines configuration 52f823c repeated quote to show how a change moves through the process 4801b87 Merged in changes (pull request #6) 1a6a403 myquote edited online with Bitbucket 3b29606 myquote2.html edited online with Bitbucket 8b236d9 myquote edited online with Bitbucket 235b9a7 testing prs c5826da more changes 43a87f4 remivng d5c4c62 a few small changes 23a7476 Merged in new-feature2 (pull request #3) 5cc4e1e add a commit message cbbb5d6 trying a thing 438f956 adding section for permissions and cleaning up some formatting 23251c1 updated snipptes.xml organization into resources. other files misc changes 3f630f8 Adding file to track changes e52470d README.md edited online with Bitbucket e2fad94 README.md edited online with Bitbucket 592f84f Merge branch 'master' into new-feature2 Merge branch especially if it merges an updated upstream into a topic branch. 7d0bab8 added a line : 여기서 우리는 repo의 커밋 내역이 이전 버전으로 복원되었음을 알 수 있습니다. 우리는 4801b87이 첫 번째 재설정 작업에서 손실 된 것처럼 보였다고하더라도이를 복구 할 수 있음을 알 수 있습니다. git reflog는 저장소의 변경 사항을 취소하기위한 강력한 도구입니다. 자세한 내용은 git reflog 페이지에서 자세히 알아보십시오. Git 가이드

2020-04-22 06:47:14.668

Git Reflog

Git은 reference log를 이용해 branch의 최신 버전을 유지한다. Reflog는 reference logs 를 뜻하며 reference 또는 ref는 commit 를 가르킨다. 기본 기능 Git 이력 정보를 얻을 수 있다. git reflog [show HEAD] 결과: eff544f HEAD@{0}: commit: migrate existing content bf871fd HEAD@{1}: commit: Add Git Reflog outline 9a4491f HEAD@{2}: checkout: moving from master to git_reflog 9a4491f HEAD@{3}: checkout: moving from Git_Config to master 39b159a HEAD@{4}: commit: expand on git context 9b3aa71 HEAD@{5}: commit: more color clarification f34388b HEAD@{6}: commit: expand on color support 9962aed HEAD@{7}: commit: a git editor -> the Git editor 특정 Reflog 보기 전체 Reflog 보기 git reflog show --all 특정 Reflog 보기 git reflog show <Reflog name> otherbranch 브랜치 지정 결과: git reflog show otherbranch 9a4491f otherbranch@{0}: commit: seperate articles into branch PRs 35aee4a otherbranch{1}: commit (initial): initial commit add git-init and setting-up-a-repo docs 시간 이력 Reflog는 timestamp를 가지고 있으므로 시간을 활용한 다양한 기능을 이용할 수 있다. master 브랜치에 대해 하루 이전의 이력 표시 예: git diff master@{0} master@{1.day.ago} 시간 표현 방법) 1.minute.ago 1.hour.ago 1.day.ago yesterday 1.week.ago 1.month.ago 1.year.ago 2011-05-17.09:00:00 Git 가이드

2020-04-22 06:06:07.517

Git Commit --amend

2014 - 2023 CURVC Corp. All rights reserved

git commit --amend는 최근 커밋을 변경하는 가장 쉬운 방법이다. 단, 커밋의 snapshot을 변경하지 않으며 커밋 메시지, 커밋 파일 추가 기능을 제공한다. 추가 옵션 없이 명령을 실행하면 편집기를 통해 커밋 메시지를 수정할 수 있다. git commit --amend -m은 편집기를 실행하지 않고 커밋 메시지 직접 입력할 때 유용하다. git commit --amend -m "커밋 메시지 수정" 또한 커밋에 누락된 파일을 추가할 수 있다. 1) A.java 수정 후 add > commit 2) B.java 수정 git commit --amend --no-edit 결과: B.java가 최종 커밋에 포함됨 Screen Shot 2020-04-22 at 6.05.46 AM.png Git 가이드

2020-04-22 06:30:21.47

Git Rebase

Rebase는 커밋을 새로운 부모 커밋으로 옮기는 기능을 제공한다. Push된 history를 rebase 하지 않도록 주의한다. 이 경우 history를 기반으로 작업하고있는 개발자들에게 영향을 주게된다. 다음 명령은 현재 working branch의 커밋을 <base> branch로 옮긴다. git rebase <base> git rebase의 기본 동작은 non-interactive이며, interactive mode (--interactive, -i)로 editor를 이용할 수 있다. git rebase -i new-base-branch editor 화면: pick 2231360 some old commit pick ee2adc2 Adds new feature # Rebase 2cf755d..ee2adc2 onto 2cf755d (9 commands) # # Commands: # p, pick = use commit (커밋 사용) # r, reword = use commit, but edit the commit message (커밋 메시지 변경) # e, edit = use commit, but stop for amending (커밋 수정을 위해 자동 수행 멈춤) # s, squash = use commit, but meld into previous commit (이전 커밋에 포함) # f, fixup = like "squash", but discard this commit's log message (이전 커밋에 포함) # x, exec = run command (the rest of the line) using shell (명령 수행) # d, drop = remove commit (커밋 삭제) editor를 종료하면 선택한 명령이 일괄 수행된다. --on-to 옵션은 브랜치을 rebase 하는 기능을 제공한다. git rebase --onto <newbase> <oldbase> featureB branch는 featureA와 의존성의 없어서 master로 rebase하고 싶은 경우, o---o---o---o---o master ₩ o---o---o---o---o featureA ₩ o---o---o featureB git rebase --onto master featureA featureB rebase 결과: o---o---o featureB / o---o---o---o---o master ₩ o---o---o---o---o featureA Screen Shot 2020-04-22 at 6.07.26 AM.png Git 가이드

2019-06-15 16:26:20.695

Git Clone

2014 - 2023 CURVC Corp. All rights reserved

git clone 명령을 자세히 살펴 보겠습니다. git clone은 기존 저장소를 대상으로하고 대상 저장소의 복제본 또는 복사본을 만드는 데 사용되는 Git 명령 줄 유틸리티입니다. 이 페이지에서는 확장 구성 옵션과 git clone의 일반적인 사용 사례에 대해 설명합니다. 여기서 다루는 몇 가지 사항은 다음과 같습니다. 로컬 또는 원격 저장소 복제 원시 저장소 복제 얕은 옵션을 사용하여 저장소를 부분적으로 복제 URL 구문 및 지원 프로토콜 이 페이지는보다 복잡한 복제 및 구성 시나리오를 탐색합니다. 목적: 저장소 간 협업 (작업을 위한 저장소 복제) 프로젝트가 중앙 저장소에 이미 설정된 경우 git clone 명령은 사용자가 개발 복사본을 얻는 가장 일반적인 방법입니다. git init과 마찬가지로, clone은 일반적으로 일회성 작업입니다. 개발자가 작업 복사본을 얻으면 모든 버전 제어 작업과 공동 작업이 로컬 저장소를 통해 관리됩니다. 저장소 간 협업 Git의 "working copy"에 대한 아이디어가 SVN 저장소에서 코드를 체크 아웃하여 얻은 작업 카피와 매우 다르다는 것을 이해하는 것이 중요합니다. SVN과는 달리, Git은 작업 카피와 중앙 저장소를 구별하지 않는다. 그것들은 모두 본격적인 Git 저장소이다. 이것은 Git과 SVN과 근본적으로 다른 협력을 만든다. SVN이 중앙 저장소와 작업 사본 사이의 관계에 의존하는 반면, Git의 공동 작업 모델은 저장소와 저장소 간의 상호 작용을 기반으로합니다. 작업 복사본을 SVN의 중앙 저장소로 검사하는 대신 한 저장소에서 다른 저장소로 커밋을 push 하거나 pull 합니다. 03.png 04.png 물론, Git repos라는 특별한 의미를주는 것을 막을 수있는 방법은 없습니다. 예를 들어 Git repo를 "중앙"저장소로 지정하면 Git을 사용하여 중앙 집중식 워크 플로를 복제할 수 있습니다. 요컨대, 이것은 VCS 자체에 내장되어 있지 않고 규칙을 통해 수행됩니다. 사용법 git clone은 주로 기존 repo를 가리키고 다른 위치의 새 디렉토리에서 해당 repo의 복제본 또는 복사본을 만드는 데 주로 사용됩니다. 원래 저장소는 로컬 파일 시스템이나 원격 머신이 액세스 할 수있는 지원 프로토콜에 위치 할 수 있습니다. git clone 명령은 기존 Git 저장소를 복사합니다. 이것은 일종의 "작업 카피 (working copy)"가 본래의 저장소를 가지고 있다는 것을 제외하고는 SVN 체크 아웃과 비슷합니다. 그것은 자체 히스토리를 가지고 있고, 자체 파일을 관리하며, 원래 저장소와 완전히 격리 된 환경입니다. 편의상 복제는 원래 저장소로 다시 향하는 "origin"이라는 원격 연결을 자동으로 만듭니다. 따라서 중앙 저장소와 쉽게 상호 작용할 수 있습니다. 이 자동 연결은 refs / remotes / origin에서 원격 지점 HEAD에 Git 참조를 만들고 remote.origin.url 및 remote.origin.fetch 구성 변수를 초기화하여 설정됩니다. git clone 사용 예제는 저장소 가이드 설정에서 찾을 수 있습니다. 아래 예제는 SSH 사용자 이름 john을 사용하여 example.com http://example.com에 액세스 할 수있는 서버에 저장된 중앙 저장소의 로컬 복사본을 얻는 방법을 보여줍니다. git clone ssh://john@example.com/path/to/my-project.git cd my-project # Start working on the project 첫 번째 명령은 로컬 컴퓨터의 my-project 폴더에있는 새 Git 저장소를 초기화하고 중앙 저장소의 내용으로 채 웁니다. 그런 다음 프로젝트에 들어가서 파일 편집, 스냅 샷 작성 및 다른 저장소와의 상호 작용을 시작할 수 있습니다. 또한 .git 확장자는 복제 된 저장소에서 생략됩니다. 이것은 로컬 복사본의 베어 메어가 아닌 상태를 반영합니다. 폴더에 복제하기 git clone <repo> <directory> <repo>에있는 저장소를 ~ <directory>라는 폴더에 복제하십시오! 로컬 컴퓨터에서. 복제 대상 테그 설정하기 git clone --branch <tag> <repo> <repo>에있는 저장소를 복제하고 <tag>에 대한 참조 만 복제하십시오. Shallow clone git clone -depth=1 <repo> <repo>에있는 저장소를 복제하고 옵션 깊이 = 1로 지정된 커밋의 이력 이 예에서는 <repo>의 복제가 만들어지고 가장 최근의 커밋만 새 복제 된 Repo에 포함됩니다. 얕은 복제는 광범위한 커밋 내역이있는 repos로 작업 할 때 가장 유용합니다. 광범위한 커밋 내역으로 인해 디스크 공간 사용 제한 및 복제시 긴 대기 시간과 같은 확장 문제가 발생할 수 있습니다. 얕은 복제는 이러한 확장 문제를 완화하는 데 도움이 될 수 있습니다. 구성 옵션 git clone -branch - branch 인수를 사용하면 원격 HEAD가 가리키는 분기 (대개 마스터 분기) 대신 복제할 특정 분기를 지정할 수 있습니다. 또한 동일한 효과를 위해 분기 대신 태그를 전달할 수 있습니다. git clone -branch new_feature git://remoterepository.git 위의 예제는 원격 Git 저장소에서 new_feature 분기 만 복제합니다. 이는 저장소의 HEAD ref 파일을 다운로드하고 필요한 ref를 추가로 가져와야하는 시간을 절약 할 수있는 순전히 유틸리티입니다. git clone -mirror vs. git clone -bare git clone --bare git init --bare와 비슷하게 -bare 인수가 git clone에 전달되면 원격 저장소의 복사본은 생략 된 작업 디렉토리로 만들어집니다. 즉, 리포지토리는 푸시 및 푸시가 가능하지만 직접 편집 할 수없는 프로젝트의 기록으로 설정됩니다. 또한 repo에 대한 원격 브랜치는 -bare 저장소로 구성되지 않습니다. git init --bare와 마찬가지로 개발자가 직접 편집하지 않는 호스트 저장소를 만드는 데 사용됩니다. git clone --mirror --mirror 인수를 전달하면 암시 적으로 --bare 인수도 전달됩니다. 즉, --bare의 동작은 --mirror에 의해 상속됩니다. 편집 가능한 작업 파일이없는 맨 페이지가 생성되었습니다. 또한 --mirror는 원격 저장소의 모든 확장 된 참조를 복제하고 원격 분기 추적 구성을 유지 관리합니다. 그런 다음 미러에서 git remote update를 실행하면 origin repo의 모든 참조를 덮어 씁니다. 정확한 '미러링 된'기능을 제공합니다. 기타 구성 옵션 다른 git clone 옵션의 포괄적 인 목록은 Git 공식 문서 https://git-scm.com/docs/git-clone를 참조하십시오. 이 문서에서는 다른 일반적인 옵션에 대해서도 다룰 것입니다. git clone --template git clone --template=<template_directory> <repo location> <repo location>에서 repo를 복제하고 <template directory>의 템플릿을 새로 만든 로컬 분기에 적용합니다. Git 템플릿에 대한 철저한 언급은 git init 페이지에서 찾을 수 있습니다. Git URLs Git은 원격 저장소 위치를 Git 명령에 전달하는 데 사용되는 자체 URL 구문을 가지고 있습니다. git clone은 원격 저장소에서 가장 일반적으로 사용되기 때문에 여기에서 Git URL 구문을 검사합니다. Git URL protocols -SSH SSH (Secure Shell)는 유비쿼터스 인증 네트워크 프로토콜로 대부분의 서버에서 기본적으로 공통적으로 구성됩니다. SSH는 인증 된 프로토콜이므로 연결하기 전에 호스팅 서버에 자격 증명을 설정해야합니다. ssh : // [사용자 @] host.xz [: 포트] /path/to/repo.git/ - GIT 자식에게 유일한 프로토콜. 자식은 포트 (9418)에서 실행되는 데몬과 함께 제공됩니다. 프로토콜은 SSH와 유사하지만 인증이 없습니다. git : //host.xz [: port] /path/to/repo.git/ - HTTP 하이퍼 텍스트 전송 프로토콜. 웹의 프로토콜로 인터넷을 통해 웹 페이지 HTML 데이터를 전송하는 데 가장 일반적으로 사용됩니다. Git은 HTTP http [s]를 통해 통신하도록 구성 될 수 있습니다 : http[s]://host.xz [: port] /path/to/repo.git/ Git 가이드

2014 - 2023 CURVC Corp. All rights reserved

2019-06-15 16:26:55.5

Git Init

이 페이지에서는 git init 명령을 자세히 살펴 봅니다. 이 페이지의 끝 부분에서 git init의 핵심 기능과 확장 기능 세트에 대해 알게 될 것입니다. 이 탐험은 다음을 포함한다 : git init 옵션과 사용법 .git 디렉토리 개요 사용자 정의 git init 디렉토리 환경 값 git init 과 git clone git init bare 저장소 git init 템플릿 git init 명령은 새로운 Git 저장소를 만듭니다. 그것은 버전없는 기존 프로젝트를 Git 저장소로 변환하거나 새로운 빈 저장소를 초기화하는 데 사용할 수 있습니다. 다른 Git 명령은 초기화 된 저장소 외부에서 사용할 수 없으므로 일반적으로 새 프로젝트에서 실행할 첫 번째 명령입니다. git init을 실행하면 현재 작업 디렉토리에 .git 하위 디렉토리가 생성됩니다.이 디렉토리에는 새 저장소에 필요한 모든 Git 메타 데이터가 들어 있습니다. 이 메타 데이터에는 개체, 참조 및 템플릿 파일의 하위 디렉터리가 포함됩니다. 현재 체크 아웃 된 커밋을 가리키는 HEAD 파일도 생성됩니다. 프로젝트의 루트 디렉토리에서 .git 디렉토리를 제외하고 기존 프로젝트는 변경되지 않습니다 (SVN과 달리 모든 하위 디렉토리에는 .git 서브 디렉토리가 필요하지 않습니다). 기본적으로 git init은 Git 구성을 .git 하위 디렉토리 경로로 초기화합니다. 다른 곳에서 살기를 원하면 하위 디렉토리 경로를 수정하고 사용자 정의 할 수 있습니다. $ GIT_DIR 환경 변수를 커스텀 경로로 설정하면 git init은 거기에있는 Git 설정 파일을 초기화 할 것이다. 또한 동일한 결과에 대해 --separate-git-dir 인수를 전달할 수 있습니다. 별도의 .git 하위 디렉토리를 사용하는 일반적인 경우는 .git 폴더를 다른 위치에 유지하면서 홈 디렉토리에 시스템 구성 "dotfiles"(.bashrc, .vimrc 등)을 유지하는 것입니다. 사용법 SVN과 비교하여 git init 명령은 새로운 버전 제어 프로젝트를 생성하는 매우 쉬운 방법입니다. Git에서는 저장소를 만들고, 파일을 가져오고, 작업 복사본을 체크 아웃 할 필요가 없다. 또한, 힘내는 기존의 서버 또는 관리자 권한이 필요하지 않습니다. 당신이해야 할 일은 프로젝트 서브 디렉토리에 cd하고 git init을 실행하는 것입니다. 여러분은 Git 저장소를 완벽하게 사용할 수 있습니다. git init 현재 디렉토리를 Git 저장소로 변환하십시오. 이렇게하면 .git 서브 디렉토리가 현재 디렉토리에 추가되고 프로젝트의 개정판을 기록 할 수있게됩니다. git init <directory> 지정된 디렉토리에 빈 Git 저장소를 만듭니다. 이 명령을 실행하면 .git 서브 디렉토리 만 포함하는 새 서브 디렉토리가 작성됩니다. 이미 프로젝트 디렉토리에서 git init을 실행했고 .git 서브 디렉토리가 있다면, 동일한 프로젝트 디렉토리에서 git init을 다시 안전하게 실행할 수 있습니다. 기존 .git 구성을 재정의하지 않습니다. git init vs. git clone 빠른 참고 사항 : git init 및 git clone은 쉽게 혼동 될 수 있습니다. 높은 수준에서, 그들은 둘 다 "새로운 자식 저장소를 초기화하는 데 사용할 수 있습니다." 그러나 git clone은 git init에 의존합니다. git clone은 기존 저장소의 복사본을 만드는 데 사용됩니다. 내부적으로, git clone은 먼저 git init을 호출하여 새로운 저장소를 만듭니다. 그런 다음 기존 저장소의 데이터를 복사하고 새로운 작업 파일 세트를 체크 아웃합니다. git clone 페이지에 대해 자세히 알아보십시오. Bare repositories --- git init --bare git init --bare <directory> 빈 저장소를 초기화하고 작업 디렉토리를 생략하십시오. 공유 저장소는 항상 --bare 플래그를 사용하여 만들어야합니다 (아래 토론 참조). 일반적으로, --bare 플래그로 초기화 된 저장소는 .git로 끝납니다. 예를 들어, my-project라는 저장소의 베어 버전은 my-project.git라는 디렉토리에 저장되어야합니다. --bare 플래그는 작업 디렉토리가없는 저장소를 작성하므로 파일을 편집하고 해당 저장소의 변경 사항을 커밋 할 수 없습니다. git push와 git pull을위한 맨 처음 저장소를 만들지 만 직접적으로 커밋하지 마십시오. 중앙 저장소는 베어지 저장소로 분기를 푸는 것이 변경 내용을 덮어 쓸 가능성이 있기 때문에 항상 베어 리지 저장소로 만들어야합니다. 개발 환경과는 반대로 저장소를 저장소 기능으로 표시하는 방법으로 생각하십시오. 즉, 사실상 모든 Git 워크 플로우에서 중앙 저장소는 노출되지 않았으며 개발자 로컬 저장소는 노출되지 않았습니다. 01.png git init --bare의 가장 일반적인 사용 사례는 원격 중앙 저장소를 만드는 것입니다 : ssh <user>@<host> cd path/above/repo git init --bare my-project.git 먼저, 중앙 저장소를 포함 할 서버로 SSH합니다. 그런 다음 프로젝트를 저장하려는 위치로 이동합니다. 마지막으로 --bare 플래그를 사용하여 중앙 저장소 저장소를 만듭니다. 개발자는 my-project.git을 복제하여 개발 컴퓨터에서 로컬 복사본을 만듭니다. git init templates git init <directory> --template=<template_directory> 새 Git 저장소를 초기화하고 <template_directory>의 파일을 저장소에 복사합니다. 템플릿을 사용하면 미리 정의 된 .git 하위 디렉토리로 새 저장소를 초기화 할 수 있습니다. 새 저장소의 .git 서브 디렉토리에 복사 할 기본 디렉토리와 파일을 갖도록 템플리트를 구성 할 수 있습니다. 디폴트의 Git 템플릿은 일반적으로`/ usr / share / git-core / templates` 디렉토리에 있지만 여러분 머신의 다른 경로 일 수 있습니다. 기본 템플리트는 좋은 참조이며 템플리트 기능을 사용하는 방법의 예입니다. 기본 템플릿에 표시되는 템플릿의 강력한 기능은 Git Hook 구성입니다. 미리 정의 된 Git 훅을 사용하여 템플릿을 만들 수 있으며, 일반적인 훅을 사용할 준비가 된 새로운 git 리포지토리를 초기화 할 수 있습니다. Git Hook에 대한 자세한 내용은 Git Hook 페이지를 참조하십시오. 구성 git init <directory>의 모든 설정은 <directory> 인수를 취합니다. <directory>를 제공하면 명령이 그 안에 실행됩니다. 이 디렉토리가 존재하지 않으면 생성됩니다. 이미 논의 된 옵션과 설정 외에도 Git init에는 몇 가지 다른 명령 행 옵션이 있습니다. 전체 목록은 다음과 같습니다. -Q, --QUIET :"위험 수준"메시지, 오류 및 경고 만 인쇄합니다. 다른 모든 출력은 음소거됩니다. --BARE :bare 저장소를 만듭니다. 위의 "bare repository"섹션을 참조하십시오. --TEMPLATE=<TEMPLATEDIRECTORY> :템플릿을 사용할 디렉토리를 지정합니다. 위의 "Git Init Templates"섹션을 참조하십시오. --SEPARATE-GIT-DIR=<GIT DIR> :<git dir>에 대한 경로를 포함하는 텍스트 파일을 생성합니다. 이 파일은 .git 디렉토리에 대한 링크 역할을합니다. .git 디렉토리를 프로젝트의 작업 파일과 별도의 위치에 저장하거나 드라이브에 저장하려는 경우

2014 - 2023 CURVC Corp. All rights reserved

유용합니다. --separate-git-dir의 일반적인 사용 사례는 다음과 같습니다. .git 폴더를 다른 위치에 유지하면서 홈 디렉토리에 시스템 구성 "dotfiles"(.bashrc, .vimrc 등)을 유지하려면 Git 히스토리는 디스크 크기가 매우 커 졌으므로 별도의 대용량 드라이브로 옮겨야합니다. Git 프로젝트를`www : root`와 같이 공개적으로 접근 가능한 디렉토리에두고 싶습니다. 기존 저장소에서 git init --separate-git-dir을 호출하면 .git 디렉토리가 지정된 <git 디렉토리> 경로로 이동합니다. --SHARED[=(FALSE|TRUE|UMASK|GROUP|ALL|WORLD|EVERYBODY|0XXX)] :새 저장소에 대한 액세스 권한을 설정하십시오. Unix 레벨 권한을 사용하는 사용자와 그룹이 저장소로 푸시 / 풀을 허용하는 사용자와 그룹을 지정합니다. Examples 기존 코드베이스를위한 새로운 git 저장소 만들기 cd /path/to/code ₩ git init ₩ git add . ₩ git commit Bare 저장소 생성하기 git init --bare /path/to/repo.git 템플릿을 이용해 저장소 생성하기 mkdir -p /path/to/template ₩ echo "Hello World" >> /absolute/path/to/template/README ₩ git init /new/repo/path --template=/absolute/path/to/template ₩ cd /new/repo/path ₩ cat /new/repo/path /README Git 가이드

2019-06-15 16:26:41.659

Git Config

이 문서에서는 git config 명령에 대해 자세히 살펴볼 것입니다. 저장소 설정 페이지에서 git config 사용법에 대해 간략히 설명했습니다. git config 명령은 전역 또는 로컬 프로젝트 수준에서 Git 구성 값을 설정하는 데 사용되는 편리한 기능입니다. 이러한 구성 수준은 .gitconfig 텍스트 파일에 해당합니다. git config를 실행하면 설정 텍스트 파일이 수정됩니다. 이메일, 사용자 이름 및 에디터와 같은 일반적인 구성 설정을 다루겠습니다. 우리는 자주 사용하는 Git 작업에 대한 바로 가기를 만들 수있는 Git 별칭에 대해 설명합니다. git config와 다양한 Git 구성 설정에 익숙해지면 강력하고 사용자 정의 된 Git 워크 플로우를 만들 수 있습니다. 사용법 git config의 가장 기본적인 사용 예는 구성 이름을 사용하여 그 이름에 설정된 값을 표시하는 것입니다. 구성 이름은 계층 구조에 따라 '섹션'과 '키'로 구성된 점으로 구분 된 문자열입니다. 예 : user.email git config user.email 이 예에서 전자 메일은 사용자 구성 블록의 하위 속성입니다. Git이 로컬에서 작성한 커밋과 연결할 전자 메일 주소를 반환합니다. git config levels and files git config 사용법을 더 자세히 설명하기 전에 잠시 시간을내어 구성 단계를 살펴 보겠습니다. git config 명령은 작동 할 구성 수준을 지정하는 인수를 허용 할 수 있습니다. 다음 구성 단계를 사용할 수 있습니다. --local 기본적으로 git config는 설정 옵션이 전달되지 않으면 로컬 레벨에 기록합니다. 로컬 레벨 설정은 git config가 호출 된 컨텍스트 저장소에 적용됩니다. 로컬 설정 값은 repo의 .git 디렉토리에있는 파일에 저장됩니다 : .git / config --global 전역 레벨 구성은 사용자별로 다르기 때문에 운영 체제 사용자에게 적용됩니다. 전역 구성 값은 사용자의 홈 디렉토리에있는 파일에 저장됩니다. ~ /.gitconfig는 유닉스 시스템에서는 C : ₩ Users ₩ <username> ₩. gitconfig in windows --system 시스템 레벨 구성은 전체 시스템에 적용됩니다. 여기에는 운영 체제의 모든 사용자와 모든 repos가 포함됩니다. 시스템 레벨 설정 파일은 시스템 루트 경로 밖의 gitconfig 파일에 있습니다. 유닉스 시스템에서는 $ (prefix) / etc / gitconfig. Windows에서이 파일은 Windows XP에서는 C : ₩ Documents and Settings ₩ All Users ₩ Application Data ₩ Git ₩ config, Windows Vista에서는 C : ₩ ProgramData ₩ Git ₩ config에 있습니다. 따라서 구성 수준의 우선 순위는 로컬, 글로벌, 시스템입니다. 이것은 구성 값을 찾을 때 힘내는 로컬 레벨에서 시작하여 시스템 레벨까지 버블 링된다는 것을 의미합니다. 변수 설정 git config에 대해 이미 알고있는 것에 대해 살펴보면서 값을 작성하는 예제를 살펴 보겠습니다. git config --global user.email "developer@curvc.com" 이 예에서는 값 developer@curvc.com mailto:developer@curvc.com을 구성 이름 user.email에 씁니다. 이 값이 현재 운영 체제 사용자에 대해 설정되도록 --global 플래그를 사용합니다. git config editor - core.editor 많은 Git 명령은 추가 입력을 요구하는 텍스트 편집기를 시작합니다. git config의 가장 일반적인 사용 사례 중 하나는 힘쓴 편집자가 사용해야하는 구성입니다. 다음은 인기있는 편집기와 일치하는 git config 명령의 표입니다. Editor config command Atom ~ git config --global core.editor "atom --wait"~ emacs ~ git config --global core.editor "emacs"~ nano ~ git config --global core.editor "nano -w"~ vim ~ git config --global core.editor "vim"~ Sublime Text (Mac) ~ git config --global core.editor "subl -n -w"~ Sublime Text (Win, 32-bit install) ~ git config --global core.editor "'c:/program files (x86)/sublime text 3/sublimetext.exe' -w"~ Sublime Text (Win, 64-bit install) ~ git config --global core.editor "'c:/program files/sublime text 3/sublimetext.exe' -w"~ Textmate ~ git config --global core.editor "mate -w"~ Merge tools 병합이 충돌하는 경우 Git은 "병합 도구"를 실행합니다. 기본적으로 Git은 일반적인 유닉스 diff 프로그램의 내부 구현을 사용한다. 내부 Git diff는 최소한의 병합 충돌 뷰어입니다. 대신 사용할 수있는 외부 타사 병합 충돌 해결 방법이 많이 있습니다. 다양한 병합 도구와 구성에 대한 개요는 Git과의 충돌을 해결하기위한 팁과 도구에 대한 가이드를 참조하십시오. git config --global merge.tool kdiff3 Colored outputs Git은 신속하게 힘내 출력을 읽는 데 도움이 색깔 터미널 출력을 지원합니다. Git 출력을 맞춤 설정하여 맞춤 설정된 색상 테마를 사용할 수 있습니다. git config 명령은이 색상 값을 설정하는 데 사용됩니다. color.ui 이것은 Git 색상의 마스터 변수입니다. false로 설정하면 Git의 컬러 터미널 출력이 모두 비활성화됩니다. git config --global color.ui false 기본적으로 color.ui는 auto로 설정되어 즉각적인 터미널 출력 스트림에 색상을 적용합니다. 자동 설정은 출력 스트림이 파일로 리디렉션되거나

2014 - 2023 CURVC Corp. All rights reserved

다른 프로세스로 파이프되는 경우 색상 코드 출력을 생략합니다. color.ui 값을 always로 설정하면 출력 스트림을 파일이나 파이프로 리디렉션 할 때 색상 코드 출력도 적용됩니다. 수신 파이프가 색으로 구분 된 입력을 기대하지 않을 수 있으므로 의도하지 않은 문제가 발생할 수 있습니다. Git color values color.ui 외에도 다른 많은 세분화 된 색상 설정이 있습니다. color.ui와 마찬가지로이 색상 설정은 모두 false, auto 또는 always로 설정할 수 있습니다. 이 색상 설정에는 특정 색상 값 세트가있을 수도 있습니다. 지원되는 색상 값의 예는 다음과 같습니다. normal black red green yellow blue magenta cyan white 색상은 # ff0000과 같은 16 진수 색상 코드 또는 터미널에서 지원하는 경우 ANSI 256 색상 값으로 지정할 수도 있습니다. Git color configuration settings 1. color.branch Git 분기 명령의 출력 색상을 구성합니다. 2. color.branch.<slot> 이 값은 Git 브랜치 출력에도 적용됩니다. <slot>은 다음 중 하나입니다. current : 현재 브랜치 local : 로컬 브랜치 remote : 원격 지점 참조 ref/remotes upstream : 업스트림 tracking branch plain : 다른 ref 3. color.diff git diff, git log 및 git show 출력에 색상 적용 4. color.diff.<slot> color.diff 아래에 <slot> 값을 설정하면 패치의 어느 부분에 특정 색상을 사용할지 알려줍니다. context : diff의 문맥 텍스트. 힘내 맥락은 변화를 강조하는 diff 또는 패치에 표시된 텍스트 내용의 줄입니다. plain : 문맥의 동의어 meta : diff의 메타 정보에 색상을 적용합니다. frag : 색을 "hunk header"또는 "hunk header의 기능"에 적용합니다. old : diff에서 제거 된 선에 색상을 적용합니다. new : diff의 추가 된 라인을 색칠합니다. 커밋 (commit) : diff에서 헤더 색상을 커밋합니다. 공백 : diff에있는 공백 오류의 색상을 설정합니다. 5. color.decorate.<slot> git log --decorate 출력의 색상을 사용자 정의하십시오. 지원되는 <slot> 값은 branch, remoteBranch, tag, stash 또는 HEAD입니다. 로컬 브랜치, 원격 추적 분기, 태그, 숨겨진 변경 사항 및 HEAD에 각각 적용 가능합니다. 6. color.grep git grep의 출력에 색상을 적용합니다. 7. color.grep.<slot> git grep에도 적용됩니다. <slot> 변수는 grep 출력의 어느 부분에 색상을 적용 할지를 지정합니다. 컨텍스트 : 컨텍스트 라인에서 일치하지 않는 텍스트 filename : 파일 이름 접두사 함수 : 함수 이름 행 linenumber : 줄 번호 접두사 일치 : 일치하는 텍스트 matchContext : 컨텍스트 라인에서 일치하는 텍스트 matchSelected : 선택한 줄의 일치하는 텍스트 selected : 선택한 줄에서 일치하지 않는 텍스트 구분 기호 : 한 줄에있는 필드 사이의 구분 기호 (:, - 및 =) 및 헌크 (-) 사이의 구분 기호 8. color.interactive 이 변수는 대화 형 프롬프트 및 표시에 색상을 적용합니다. 예는 git add --interactive 및 git clean입니다. --interactive 9. color.interactive.<slot> 더 구체적인 "대화 형 출력"을 목표로 지정할 수있는 <slot> 변수가 있습니다. 사용 가능한 <slot> 값은 prompt, header, help, error입니다. 각각은 해당 대화식 출력에서 작동합니다. 10. color.pager 호출기가 사용 중일 때 유색 출력을 활성화 또는 비활성화합니다. 11. color.showBranch git show branch 명령의 컬러 출력을 활성화 또는 비활성화합니다. 12. color.status Git 상태에 대한 색상 출력을 활성화 또는 비활성화하는 부울 값입니다. 13. color.status.<slot> 지정된 자식 상태 요소에 대한 맞춤 색상을 지정하는 데 사용됩니다. <slot>은 다음 값을 지원합니다. header 상태 영역의 헤더 텍스트를 대상으로합니다. added or updated 추가되었지만 커밋되지 않은 두 대상 파일 changed 수정되었지만 git 인덱스에 추가되지 않은 파일을 대상으로합니다. untracked Git에 의해 추적되지 않는 파일을 대상으로합니다. branch 현재 분기에 색상을 적용합니다. nobranch "분기 없음"경고가 표시된 색상 unmerged 변경 사항이 병합되지 않은 색상 파일 Aliases 운영 체제 명령 줄에서 별칭 개념을 잘 알고있을 수 있습니다. 그렇지 않은 경우, 더 긴 명령이나 결합 된 명령으로 확장 할 명령을 정의하는 사용자 지정 바로 가기입니다. 별칭을 사용하면 자주 사용하는 명령을 입력하는 데 드는 시간과 에너지 비용을 절약 할 수 있습니다. 힘내 자체 별칭 시스템을 제공합니다. Git 별칭의 일반적인 사용 예는 커밋 명령을 줄이는 것입니다. 힘내 별칭은 힘내 설정 파일에 저장되어있다. 즉, git config 명령을 사용하여 별칭을 구성 할 수 있습니다. git config --global alias.ci commit 이 예는 git commit 명령에 대한 ci 별명을 작성합니다. 그런 다음 git ci를 실행하여 git commit을 호출 할 수 있습니다. 별칭은 다른 별칭을 참조하여 강력한 콤보를 만들 수도 있습니다. git config --global alias.amend ci --amend 이 예제는 ci 별명을 --amend flag를 사용하는 새 별명으로 작성하는 별명 수정을 작성합니다. Formatting & whitespace Git은 git diff를 사용할 때 공백 문제를 강조 표시하도록 구성 할 수있는 여러 "공백"기능을 가지고 있습니다. 공백 문제는 구성된 색상 color.diff.whitespace를 사용하여 강조 표시됩니다. 기본적으로 사용 가능한 기능은 다음과 같습니다. blank-at-eol :하이라이트 라인 엔딩에서 고아 공백 space-before-tab :줄을 들여 쓰기 할 때 탭 문자 앞에 나타나는 공백 문자를 강조 표시합니다. blank-at-eof :파일 끝에 삽입 된 빈 줄을 강조합니다. 다음 기능은 기본적으로 사용하지 않도록 설정되어 있습니다. indent-with-non-tab: 탭 대신 공백으로 들여 쓰는 줄을 강조 표시합니다. tab-in-indent :초기 탭 들여 쓰기를 오류로 강조 표시합니다. tailing-space :blank-at-eol과 blank-at-eof 모두에 대한 줄임말입니다. cr-at-eol :줄 끝에서 캐리지 리턴을 강조 표시합니다. tabwidth = <n> :탭이 차지하는 문자 위치의 수를 정의합니다. 기본값은 8입니다. 허용되는 값은 1-63입니다. Git 가이드

2022-04-13 13:34:16.587

Linux CentOS 8.5 EOL -> CentOS Stream 마이그레이션(Migration

2014 - 2023 CURVC Corp. All rights reserved

가이드 제목은 [도구명] [내용]-하기 형태(ex Bitbucket Cloud 저장소 생성하기)로 입력한다. 이 문서에 대한 요약 /개요를 입력한다. (구글에서 검색되었을때 표시되는 문장) 이 문서는 Linux CentOS 8.5 EOL -> CentOS Stream 마이그레이션 (Migration) 가이드를 공유하기 위해 작성되었다. 2021-12-13 부로 지원 목록에서 제외된 Linux CentOS 8 EOL 버전에서 CentOS Stream(2024-05-31 EOL) 버전으로 마이그레이션 하는 방법을 가이드한다. 가이드에 사용되는 도구 정보를 입력한다. 도구명 Linux CentOS 버전 8.5 EOL (마지막 버전) 비고 21 complete sudo 권한 필요 23 complete 외부인터넷 필요 첨부한 이미지 너비는 800px 을 넘기지 않는다. 마이그레이션 수행 1.OS 최신 버전 업데이트 다음 명령어를 순서대로 실행한다. (최신 버전일 경우 해당 순서 생략) dnf 명령어 실행하여 No URLs in mirrorlist 이슈 발생시 Linux CentOS 8 EOL yum, dnf No URLs in mirrorlist 이슈 해결하기 1번 방법 적용 # CentOS Linux 8을 최신버전으로 업데이트 sudo dnf --refresh upgrade # 업데이트 후 재부팅 sudo reboot image2022-4-5_16-55-15.png 2.현재 OS 버전 확인 다음 명령어를 실행하여 현재 CentOS 버전을 확인한다. cat /etc/redhat-release # 더 자세히 보고 싶을 경우 다음 명령어 사용 cat /etc/os-release image2022-4-5_16-36-36.png 3.마이그레이션 수행 # CentOS Stream Repository 설치 sudo dnf install centos-release-stream -y # 기존 Repository를 CentOS Stream Repository로 교체 sudo dnf swap centos-{linux,stream}-repos -y # CentOS Linux 8 -> CentOS Stream 8 마이그레이션 및 동기화 sudo dnf distro-sync -y image2022-4-5_15-11-27.png image2022-4-5_15-54-40.png image2022-4-5_15-56-0.png 4.OS 버전 재확인 다음 명령어를 실행하여 CentOS Stream으로 변경된 OS를 확인한다. cat /etc/redhat-release # 더 자세히 보고 싶을 경우 다음 명령어 사용 cat /etc/os-release image2022-4-5_16-4-18.png 참조 CentOS Linux EOL https://www.centos.org/centos-linux-eol/ CentOS Project shifts focus to CentOS Stream https://blog.centos.org/2020/12/future-is-centos-stream/ Comparing CentOS Stream and CentOS Linux https://www.centos.org/cl-vs-cs/ How To Migrate To CentOS Stream 8 From CentOS Linux 8 - OSTechNix https://ostechnix.com/how-to-migrate-to-centos-stream-8-from-centos-linux-8/

2022-04-13 13:34:25.761

Linux CentOS 8.5 EOL -> Rocky Linux 8 마이그레이션(Migration)

가이드 제목은 [도구명] [내용]-하기 형태(ex Bitbucket Cloud 저장소 생성하기)로 입력한다. 이 문서에 대한 요약 /개요를 입력한다. (구글에서 검색되었을때 표시되는 문장) 이 문서는 Linux CentOS 8.5 EOL -> Rocky Linux 8 마이그레이션 (Migration) 가이드를 공유하기 위해 작성되었다. 2021-12-13 부로 지원 목록에서 제외된 Linux CentOS 8 EOL 버전에서 Rocky Linux 8 버전으로 마이그레이션 하는 방법을 가이드한다. 가이드에 사용되는 도구 정보를 입력한다. 도구명 Linux CentOS 버전 8.5 EOL (마지막 버전) 비고 21 complete sudo 권한 필요 41 complete 외부 인터넷 연결 필요 첨부한 이미지 너비는 800px 을 넘기지 않는다. 마이그레이션 수행 1.OS 최신 버전 업데이트 다음 명령어를 순서대로 실행한다. (최신 버전일 경우 해당 순서 생략) dnf 명령어 실행하여 No URLs in mirrorlist 이슈 발생시 Linux CentOS 8 EOL yum, dnf No URLs in mirrorlist 이슈 해결하기 1번 방법 적용 # CentOS Linux 8을 최신버전으로 업데이트 sudo dnf --refresh upgrade # 업데이트 후 재부팅 sudo reboot image2022-4-5_16-55-31.png 2.현재 OS 버전 확인 다음 명령어를 실행하여 현재 CentOS 버전을 확인한다. cat /etc/redhat-release # 더 자세히 보고 싶을 경우 다음 명령어 사용 cat /etc/os-release image2022-4-5_16-36-36.png 3.마이그레이션 수행 다음 명령어를 순서대로 실행하여 Rocky Linux 마이그레이션 스크립트를 다운로드 후 실행 한다. # 스크립트 다운로드 (스크립트 정보는 아래 참조 URL github에서 확인) curl -O https://raw.githubusercontent.com/rocky-linux/rocky-tools/main/migrate2rocky/migrate2rocky.sh # 스크립트 실행 권한 부여 sudo chmod +x migrate2rocky.sh # 스크립트 실행 sudo bash migrate2rocky.sh -r 스크립트 실행시 다음과 같이 수행된다. CentOS 8 Repositories → Rocky Linux Repositories 교체 CentOS Branding 제거 Rockey Linux GPG 키 추가 모든 패키지 업그레이드 image2022-4-5_17-2-20.png 스크립트 실행이 완료되면 다음과 같이 표시된다. image2022-4-5_17-22-3.png 3.동기화 다음 명령어를 실행하여 설치된 패키지를 사용 가능한 최신 버전으로 동기화한다. sudo dnf distro-sync -y image2022-4-5_17-37-56.png 4.시스템 재시작 다음 명령어를 실행하여 시스템을 재시작한다. sudo reboot 5.OS 버전 재확인 다음 명령어를 실행하여 Rockey Linux로 변경된 OS를 확인한다. cat /etc/redhat-release image2022-4-5_17-39-10.png 참조 Rocky Linux https://rockylinux.org/ How To Migrate To Rocky Linux 8 From CentOS 8 Linux (ostechnix.com) https://ostechnix.com/how-to-migrate-to-rocky-linux-8-from-centos-8-linux/ rocky-tools/migrate2rocky at main · rocky-linux/rocky-tools · GitHub https://github.com/rocky-linux/rocky-tools/tree/main/migrate2rocky

2014 - 2023 CURVC Corp. All rights reserved

2022-04-13 13:34:34.515

Linux CentOS 8 EOL Yum, Dnf No URLs In Mirrorlist 이슈 해결하기

가이드 제목은 [도구명] [내용]-하기 형태(ex Bitbucket Cloud 저장소 생성하기)로 입력한다. 이 문서에 대한 요약 /개요를 입력한다. (구글에서 검색되었을때 표시되는 문장) 이 문서는 Linux CentOS 8 EOS yum, dnf No URLs in mirrorlist 이슈 해결 가이드를 공유하기 위해 작성되었다. No URLs in mirrorlist 이슈는 CentOS 8 EOS 버전에서 yum 또는 dnf 명령어를 사용하여 패키지를 설치하거나 업데이트를 수행 할 때 발생 할 수 있다. 가이드에 사용되는 도구 정보를 입력한다. 도구명 Linux CentOS 버전 8.5 EOL (마지막 버전) 비고 21 complete sudo 권한 필요 첨부한 이미지 너비는 800px 을 넘기지 않는다. 원인 파악 1.오류 확인 yum update 또는 dnf install 등의 명령어를 실행 할 때 다음과 같이 오류가 발생한다. Error: Failed to download metadata for repo 'appstream': Cannot prepare internal mirrorlist: No URLs in mirrorlist image2022-4-5_16-16-58.png 2.오류 원인 해당 오류는 Centos 패키지 저장소의 주소가 잘못되거나, 접속하지 못할 때 발생하는 오류로, 공식 지원이 끝난 CentOS 8에서 빈번하게 발생할 수 있다. 해결 방법 다음 방법 중 하나의 방법을 사용하여 해결 가능하다. 1번 해결 방법은 공식 지원이 끝난 OS에 대해 일시적인 방법으로, OS를 마이그레이션 하는 2번 해결 방법을 권장한다. 1.저장소 경로 변경 다음 명령어를 실행하여 저장소 경로를 변경한다. sudo sed -i -e "s|mirrorlist=|#mirrorlist=|g" /etc/yum.repos.d/CentOS-* sudo sed -i -e "s|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g" /etc/yum.repos.d/CentOS-* 2.OS 마이그레이션 다음 두 방법 중 한가지를 사용하여 OS를 마이그레이션한다. Linux CentOS 8.5 EOL -> CentOS Stream 마이그레이션 Linux CentOS 8.5 EOL → Rocky Linux 마이그레이션

2023-06-22 10:48:18.666

Linux 명령어

이 문서는 Linux 명령어에 대한 정보를 공유하기 위해 작성되었다.

2023-06-23 09:18:46.263

리눅스 Top 명령어

이 문서는 리눅스 top 명령어에 대한 정보를 공유하기 위해 작성되었다. top 명령어는 CPU, Memery 등 시스템의 전반적인 상태를 파악할 수 있게 해주는 명령어로 실행 시, 기본 3초 간격으로 화면을 갱신하여 보여준다. top 주요 정보 11:37:26 up : top 정보 제공 시간 141 days : 서버 구동 시간 2 users : 접속 사용자 수 load average : 3개 숫자 차례대로 1분, 5분, 15분 간의 평균 실행/대기 중인 프로세스의 수를 나타냄 (CPU 코어수 보다 적으면 문제 없음) Tasks : 프로세스 개수 %Cpu(s) : CPU 정보 MiB Mem, Swap : 메모리 사용량 top 프로세스 목록 정보 PID : 프로세스 ID USER : 실행 유저 PR : 프로세스 Priority 값 - 0 최고순위, 20, 일반적인 우선순위, 39 - 최하우선순위 NI : 프로세스 Nice VIRT : 프로세스가 할당된 가상 메모리 전체 (단위 KiB, 실제로 할당되지 않은 가상 공간으로 물리 메모리를 의미하지 않음. 해당 값이 크다고 해도 큰 문제는 없음) RES : 현재 프로세스가 사용하고 있는 물리 메모리 (단위 KiB) SHR : 다른 프로세스와 공유하고 있는 메모리 (단위 KiB) S : 프로세스 상태 D = 무중단 Sleep 상태 (Uninterruptible Sleep) I = Idle R = Running S = Sleeping T = 작업 제어 신호(Job Control Signal)에 의해 중지됨 t = Trace 중 디버거에 의해 중지됨 Z = Zombie %CPU %MEM : RAM에서 RES가 차지하는 비율 TIME+ : 프로세스가 사용한 총 CPU 시간 COMMAND : 프로세스를 실행한 명령어 또는 관련 프로그램의 이름 image2023-6-22_11-37-48.png top 유용한 명령어 명령어 설명 1 CPU 코어별 사용 현황 m 메모리 사용률 시각화 Shift + p CPU 사용률이 높은 프로세스 나열 Shift + m 메모리 사용률이 높은 프로세스 나열 Shift + t 수행 시간이 긴 프로세스 나열 k kill 할 PID 입력 가능 H 상단의 Tasks를 기준을 쓰레드로 변경 u 모니터링 할 계정 선택하여 해당 권한 프로세스 감시

2014 - 2023 CURVC Corp. All rights reserved

2021-03-03 13:34:37.639

GitLab Offline 설치 가이드

Linux 7 설치 설치 파일 사전 준비 사항 1. 필수 사항 policycoreutils-python / openssh-server / openssh-clients / git2.29 이상 2. 선택 사항 curl / ca-certificates / postfix GitLab 설치 offline 설치 방법 (1) offline-{파일명}.tar.gz 압축 해제 (2) offline-{파일명}.repo 파일 복사 → /etc/yum.repos.d 디렉토리 (3) offline-{파일명} 디렉토리 복사 → /var/tmp 디렉토리 (4) 설치 진행 yum -y install --disablerepo=₩* --enablerepo=offline-{파일명} {파일명} -- Git 설치 $ tar -xvf offline-git-2.30.tar.gz $ cp -r git /var/tmp/ $ cp -r offline-git.repo /etc/yum.repos.d/ $ yum -y install --disablerepo=₩* --enablerepo=offline-git git -- Gitlab 설치 $ tar -xvf offline-gitlab-ee.13.9.tar.gz $ cp -r gitlab-ee /var/tmp/ $ cp -r offline-gitlab-ee.repo /etc/yum.repos.d/ $ yum -y install --disablerepo=₩* --enablerepo=offline-gitlab-ee gitlab-ee rpm 설치 디렉토리 정보 정보 디렉토리 경로 설치 디렉토리 /opt/gitlab 설정 파일 디렉토리 /etc/gitlab 데이터 디렉토리 /var/opt/gitlab 로그 디렉토리 /var/log /gitlab 포트 변경 * 포트 링크 https://docs.gitlab.com/omnibus/package-information/defaults.html https://docs.gitlab.com/omnibus/package-information/defaults.html $ vi /etc/gitlab/gitlab.rb 1.external_url 에서 포트 변경 external_url 'http://192.168.137.13:7990' 2. Advanced settings에서 주석 제거,포트 변경 puma['port'] = 8080 # 디폴트 포트가 자주 사용하는 포트이므로 확인 후 변경 Repository 디렉토리 변경 $ vi /etc/gitlab/gitlab.rb For setting up different data storing directory ! Docs: https://docs.gitlab.com/omnibus/settings/configuration.html#storing-git-data-in-an-alternative-directory ! **If you want to use a single non-default directory to store git data use a ! path that doesn't contain symlinks.** git_data_dirs({ "default" => {"path" => "/test/git-data"} # Repository 디렉토리 입력 }) 설정 변경 최초 reconfigure 실행 $ gitlab-ctl reconfigure 시작 / 종료 / 재시작 $ gitlab-ctl start $ gitlab-ctl stop $ gitlab-ctl restart GitLab 백업 방법 * 백업 링크 https://docs.gitlab.com/omnibus/settings/backups.html https://docs.gitlab.com/omnibus/settings/backups.html https://docs.gitlab.com/ee/raketasks/backup_restore.html#back-up-gitlab https://docs.gitlab.com/ee/raketasks/backup_restore.html#back-up-gitlab < config 백업 > EX)명령어 : sudo gitlab-ctl backup-etc {경로 입력} # 디폴트 디렉토리 : /etc/gitlab/config_backup $ sudo gitlab-ctl backup-etc /test/gitlab_backup <data 백업> $ sudo gitlab-backup create $ vi /etc/gitlab/gitlab.rb 설정 에서 변경 가능 -> 주석 제거,디렉토리 변경 -> gitlab-ctl reconfigure 진행 gitlab_rails['backup_path'] = "/var/opt/gitlab/backups" 백업 Crontab 설정 $ vi /opt/gitlab/bin/gitlab_cron.sh gitlab-ctl backup-etc /test /gitlab_backup gitlab-backup create $chmod +x /opt/gitlab/bin/gitlab_cron.sh $crontab -e 0 1 * * * /opt/gitlab/bin/gitlab_cron.sh # 매일 오전 1시 백업을 예약 서비스 등록 $ vi /etc/systemd/system/gitlab-run.service ========== [Unit] Description=gitlab server Service [Service] user=root ExecStart=/opt/gitlab /bin/gitlab-ctl start [Install] WantedBy=multi-user.target ========== $ systemctl enable gitlab-run.service $ systemctl daemon-reload $ systemctl start gitlab-run.service $ systemctl status gitlab-run.service

2023-04-17 09:12:00.11

Nginx SSL 인증서(https) 설정

2014 - 2023 CURVC Corp. All rights reserved

이 문서는 Nginx SSL 인증서 설정을 공유하기 위해 작성되었다. Nginx 는 일반적으로 .pem 확장자로 SSL 인증서 설정을 진행합니다. PEM 암호를 제거하는 방법 openssl rsa명령을 사용하여 암호를 제거 할 수 있습니다 . 인수로 SSL을 전달 하고 파일을 출력으로 .key얻습니다 . $ openssl rsa -in root_with_pass.key -out root.key Nginx.conf에 해당 SSL 인증서 정보를 입력합니다. crt 파일 ssl_certificate /etc/nginx/ssl/nginx.crt; ssl_certificate_key /etc/nginx/ssl/nginx.key; pem 파일 ssl_certificate root.pem; ssl_certificate_key root.key; 예시 1) Confluence server { listen www.example.com:80; server_name www.example.com; listen 443 default ssl; ssl_certificate /etc/nginx/ssl/nginx.crt; ssl_certificate_key /etc/nginx/ssl/nginx.key; ssl_session_timeout 5m; ssl_protocols TLSv1 TLSv1.1 TLSv1.2; ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20- POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256: ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA- AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA- AES256-SHA384:ECDHE-RSA-AES128-SHA: ECDHE-ECDSA-AES256-SHA384:ECDHE- ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE- RSA-DES-CBC3-SHA: EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM- SHA384: AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS'; ssl_prefer_server_ciphers on; location /synchrony { proxy_set_header X-Forwarded-Host $host; proxy_set_header X-Forwarded-Server $host; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; proxy_pass http://localhost:8091/synchrony; proxy_http_version 1.1; proxy_set_header Upgrade $http_upgrade; proxy_set_header Connection "Upgrade"; } location /confluence { client_max_body_size 100m; proxy_set_header X-Forwarded-Host $host; proxy_set_header X-Forwarded-Server $host; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; proxy_pass http://localhost:8090/confluence; } } 예시 2) Bamboo server { listen 443 ssl; server_name confluence.curvc.com; ssl_certificate root.pem; ssl_certificate_key root.key; ssl_session_timeout 5m; ssl_protocols TLSv1 TLSv1.1 TLSv1.2; ssl_ciphers ECDHE-RSA-AES128-SHA256:AES128-GCM-SHA256:!RC4:HIGH:!MD5:!aNULL:!EDH; ssl_prefer_server_ciphers on; location / { proxy_pass_header Server; proxy_set_header X-Forwarded-Host $host; proxy_set_header X-Forwarded-Server $host; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; proxy_pass https://1.1.1.10:8080; #프록시 IP } error_page 401 402 403 404 /error.html; location = /error.html { root html; } error_page 500 502 503 504 /50x.html; location = /50x.html { root html; } }

2020-10-20 15:21:36.546

Nginx Windows Service 등록

이 문서는 Nginx를 Windows Service 등록을 위한 가이드를 공유하기 위해 작성되었다. Non-Sucking Service Manager 다운로드 몇몇 방법들이 있으나, 아래가 NSSM을 통해서 진행하는 것이 가장 간단함으로 아래 경로에서 다운로드를 받는다. https://nssm.cc/download https://nssm.cc/download Nginx 서비스 등록 다운로드 받은 nssm 압축 파일을 적절한 위치에 압축을 해제하고 CMD를 통해서 해당 위치로 이동한다. 예) D:₩nssm-2.24₩win64 다음 명령을 통해 서비스 등록을 수행한다. nssm.exe install nginx 서비스 등록 UI 창이 나타나면 다음을 설정하고 하위 Install service 버튼을 클릭한다. Path : D:₩nginx₩nginx.exe Startup directory : 자동 입력됨 서비스를 실행하고 정상 동작하는지 확인한다.

2018-03-02 20:00:13.246

Private DNS 구성 (bind9)

2014 - 2023 CURVC Corp. All rights reserved

이 페이지는 내부 서버를 위한 DNS 환경 구성을 설명한다. DNS 서버는 Ubuntu를 고려하고 단일 DNS 서버를 구성한다. 패키지 설치 $ sudo apt-get install bind9 bind9utils bind9-doc /etc/default/bind9 수정 "-f -4" 추가 # startup options for the server OPTIONS="-f -4 -u bind" Options 파일 설정 options { directory " /var/cache/bind"; // ISP's domain name server forwarders { 168.126.63.1; }; dnssec-validation auto; auth-nxdomain no; # conform to RFC1035 listen-on-v6 { any; }; allow-transfer { none; }; }; Local 파일 설정 외부 domain: "curvc.com", local 대신 다른 이름 사용해도 됨 내부 IP 주소: 192.168.0.xxx로 구성된 예 zone "local.curvc.com" { type master; file "/etc/bind/zones/db.local.curvc.com"; }; zone "0.168.192.in-addr.arpa" { type master; file "/etc/bind/zones/db.192.168.0"; //192.168.0.0/24 subnet }; Forward Zone 설정 host.local.curvc.com이 요청될 대 참조하는 설정 설정이 변경할 때 Serial 숫자가 증가되어야 함 $TTL 604800 @ IN SOA ns1.local.curvc.com. admin.local.curvc.com. (4 ; Serial 604800 ; Refresh 86400 ; Retry 2419200 ; Expire 604800) ; Negative Cache TTL ; name servers - NS records IN NS ns1.local.curvc.com. ; IN NS ns2.local.curvc.com. ; name servers - A records ns1.local.curvc.com. IN A 192.168.0.104 ;ns2.local.curvc.com. IN A 192.168.0.105 ; 192.168.0.0/24 - A records ni.local.curvc.com. IN A 192.168.0.50 Reverse Zone 파일 설정 IP 주소 ("192.168.0.100")으로 DNS를 요청받으면 참조하는 설정 설정이 변경할 때 Serial 숫자가 증가되어야 함 $TTL 604800 @ IN SOA ns1.local.curvc.com. admin.local.curvc.com. (2 ; Serial 604800 ; Refresh 86400 ; Retry 2419200 ; Expire 604800) ; Negative Cache TTL ; name servers IN NS ns1.local.curvc.com. ; IN NS ns2.local.curvc.com. ; PTR records 104 IN PTR ns1.local.curvc.com. ; 192.168.0.104 ;105 IN PTR ns2.local.curvc.com. ; 192.168.0.105 50 IN PTR ni.local.curvc.com. ; 192.168.0.50 설정 확인 $ sudo named-checkconf #named.conf* 설정 확인 $ sudo named-checkzone local.curvc.com /etc/bind/zones/db.nyc3.example.com #forward zones 설정 확인 $ sudo named-checkzone 0.168.192.in-addr.arpa /etc/bind/zones/db.192.168.0 서비스 재기동 $ sudo service bind9 restart Client 설정 내부 DNS 서버 주소를 공인 DNS보다 먼저 설정한다. search local.curvc.com #local domain name nameserver 192.168.0.104 #local DNS 서버 주소 nameserver 203.248.252.2 #공인 DNS 서버 nameserver 164.124.101.2 Example (one dns server) Configuration Files bind9.tar.gz

2022-05-08 20:40:53.308

Linux에서 Non Root 권한으로 MSSQL 서버 접속하기

이 문서는 root 권한이 없는 시스템에서 MS SQL등의 데이터베이스 서버에 command line으로 접속하는 방법을 정리한다. 바이너리 준비 실제 Linux 환경에 root 권한이 없어 패키지 설치가 불가하므로 별도의 임시 서버에서 바이너리를 준비한다. 실제 구성할 서버와 동일한 OS 버전의 VM 준비 mssql tool 설치 curl https://packages.microsoft.com/config/rhel/8/prod.repo > /etc/yum.repos.d/msprod.repo 설치된 파일 묶음 만들기 - MSSQL 도구는 CentOS의 경우 /opt에 설치됨 cd /opt tar -zcvf /tmp/mssql.tar.gz mssql_tools microsoft - odbc library 묶음 만들기 cd /usr/lib64 tar -zcvf /tmp/libodbc.tar.gz libodbc.tar.gz libodbc* odbcinst binary 복사 /usr/bin/odbcinst 실제 환경 구성 임시 서버에서 준비된 바이너리를 이용해 실제 환경을 구성한다. 파일 압축 해제 mkdir ~/mssql cd ~/mssql tar -zxvf ~/Downloads/mssql.tar.gz mkdir lib64 cd lib64 tar -zxvf ~/Downloads/libodbc.tar.gz odbc driver 설정 vi ~/mssql/microsoft/msodbcsql17/etc/odbcinst.ini [ODBC Driver 17 for SQL Server] Description=Microsoft ODBC Driver 17 for SQL Server Driver=/home/username/mssql/microsoft/msodbcsql17/lib64/libmsodbcsql-17.9.so.1.1 <-- 바이너리 환경에 맞게 경로 수정 실행 환경 변수 설정 export PATH="$PATH:$HOME/mssql/mssql-tools/bin" export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$HOME/mssql/lib64" export ODBCSYSINI=$HOME/mssql/microsoft/msodbcsql17/etc 서버 접속 시험 쿼리 파일 생성 USE <database name> GO SELECT TOP (100) * FROM MY_TABLE GO 쿼리 실행 sqlcmd -U <username> -P <password> -S <SQL sever IP>,<port number> -i <쿼리 파일 이름> 예) sqlcmd -U curvc -P 비밀번호 -S 10.100.10.10,1433 -i query.sql 쿼리 결과 저장 CSV 출력 HEADER 포함 sqlcmd -U <username> -P <password> -S <SQL sever IP>,<port number> -i <쿼리 파일 이름> -W -f 65001 -s "," -o "<OutputFilePath>" HEADER 제외 sqlcmd -U <username> -P <password> -S <SQL sever IP>,<port number> -i <쿼리 파일 이름> -W -f 65001 -h -1 -s "," -o "<OutputFilePath>" 참고 https://www.easysoft.com/support/kb/kb00635.html https://www.easysoft.com/support/kb/kb00635.html

2014 - 2023 CURVC Corp. All rights reserved

2019-07-04 17:48:12.485

Private DNS 구성 (powerdns + Web)

이 문서는 powerdns / MySQL / web으로 DNS를 구성하는 방법을 제공합니다. Web으로 domain name 관리할 수 있어서 관리자의 수고가 줄어듭니다. Overview 목표 구성 내부 망에서 사용하는 도메인과 공인 도메인 혼용 지원 이를 위해 DHCP 서버 dns 설정에 내부 (private) dns 서버 추가 (primary dns: 첫 번째 dns) 고정 IP 를 사용하는 PC의 경우 dns 서버 추가 (primary dns: 첫 번째 dns) 적용 솔루션 (powerdns) PDNS는 PowerDNS BV사에서 만든 네임서버 입니다. 도메인 관리를 database를 통해 도메인 관리를 하기 때문에 대량의 domain 운영이 가능하고 캐쉬 잘못에 의한 오류 가능성이 없습니다. MySQL 설치 (PostgreSQL 도 지원) powerdns는 MySQL과 PostgreSQL 그리고 SQLite를 지원합니다. 이 문서는 MySQL database을 고려합니다. PDNS 설치 전 MySQL을 설치합니다. Database 구성 (MySQL) powerdns database 생성 powerdns database 사용자 생성 powerdns database 사용자에게 powerdns database 엑세스 권한 부여 $ mysql -uroot -p mysql > create database powerdns; mysql > GRANT ALL ON powerdns.* TO 'powerdns'@'localhost' IDENTIFIED BY 'password'; mysql > FLUSH PRIVILEGES; PDNS 설치 설치 과정에서 database 생성을 위해 미리 configuration 파일 생성 Database table 생성이 안되었다면 "/usr/share/doc /pdns-backend-mysql/README.Debian"을 참조하여 수동으로 table 생성 # MySQL Configuration file launch=gmysql gmysql-host=localhost gmysql-dbname=powerdns gmysql-user=powerdns gmysql-password=password # yum -y install pdns pdns-backend-mysql Database table 생성이 안되었다면 " /usr/share/doc/pdns-backend-mysql/README.Debian"을 참조하여 수동으로 table 생성 PDNS DB 구성 pdns backend (MySQL) 연결 설정 # MySQL Configuration file launch=gmysql gmysql-host=localhost gmysql-dbname=powerdns gmysql-user=powerdns gmysql-password=password PDNS 서비스 재시작 pdns 서비스가 자동 시작되도록 등록하고 서비스 재시작 $ chkconfig pdns on $ service pdns start PDNS 서비스 검증 dig 명령 수행 결과가 다음 예와 유사한지 확인 root@ns1:~# dig @127.0.0.1 ; <<>> DiG 9.9.5-3-Ubuntu <<>> @127.0.0.1 ; (1 server found) ;; global options: +cmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27248 ;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1 ;; WARNING: recursion requested but not available ;; OPT PSEUDOSECTION: ; EDNS: version: 0, flags:; udp: 2800 ;; QUESTION SECTION: ;. IN NS ;; Query time: 1 msec ;; SERVER: 127.0.0.1#53(127.0.0.1) ;; WHEN: Sun Nov 02 18:58:20 EST 2014 ;; MSG SIZE rcvd: 29 Apple 장치 호환성 웹 서버 설치 PowerAdmin은 web 서버를 포함하지 않고 PHP로 제작된 웹 기능을 제공하기 때문에 웹서버 (Nginx, Apache, ...)를 설치해야 한다. 이 문서는 Apache를 웹 서버로 기준으로 설명한다. $ yum -y install httpd httpd-devel php php-mcrypt php-pdo php-mysql php-devel php-gd php-imap php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring gettext wget httpd 서비스 재시작 Apache 서비스가 자동시작되도록 등록하고 재시작 한다. $ service httpd restart $ chkconfig httpd on Pear 패키지 설치 $ yum -y install php-pear-MDB2 php-pear-MDB2-Driver-mysql PowerAdmin (Web 인터페이스) 설치 powerdns의 웹 인터페이스를 제공하는 PowerAdmin을 설치한다. 다운로드 및 압축 풀기 $ cd /tmp/ $ wget https://github.com/downloads/poweradmin/poweradmin/poweradmin-2.1.6.tgz $ tar xvfz poweradmin-2.1.6.tgz $ mv /var/www/html /var/www/html.bak $ mv poweradmin-2.1.6 /var/www/html $ cd poweradmin/inc/ $ mv config-me.inc.php config.inc.php $ chown -R www-data.www-data /var/www/html Database 접속 정보 설정 Modify db_pass and session_key to your own values : $db_host = 'localhost'; $db_port = '3306'; $db_user = 'powerdns'; $db_pass = 'password'; $db_name = 'powerdns'; $db_type = 'mysql'; $session_key = 'Passw0rd'; 설정 적용을 위해 httpd 서비스 재기동 $ service httpd restart Web을 통해 PowerAdmin 설정 브라우져를 통해 http://<IP> 접속 (예: http://192.168.0.3) Click on Install -> Choose I prefer to proceed in English -> Go to Step2 -> Go to Step3 -> Username: powerdns Password: password Database Type: MySQL Hostname: localhost DB Port: 3306 Database: powerdns PowerAdmin Administrator Password: Passw0rd Go to Step4 -> This step is optional (SKIP) -> Go to Step5 -> Go to Step6 -> Go to Step7. 설치 폴더 제거 /var/www/html/install 폴더를 다른 이름으로 변경 $ mv /var/www/html/install /var/www/html/install.bak 도메인 관리 로그인 브라우져를 통해 http://<서버 IP> 접속 User name: admin Password: curvc1004 master zone 추가 Zone (domain)을 추가한다. Screen Shot 2018-03-02 at 9.45.07 PM.png reverse zone (PTR) 추가 (옵션) Problem with Record (PTR)은 IP를 기준으로 domain name을 추적할 때 사용되는 기록이다. 등록할 domain의 network (subnet)에 해당하는 zone 생성 예) 192.168.0.0/24 네트웍 이라면 → 0.168.192.in-addr.arpa 예) 10.0.0.0/8 네트웍 이라면 → 10.in-addr.arpa Screen Shot 2018-03-02 at 9.49.49 PM.png 도메인 등록 Step 1) list zone >> 원하는 zone의 연필 모양 icon 클릭 Screen Shot 2018-03-02 at 10.01.34 PM.png Step 2) 추가할 도메인 이름과 IP 주소, 그리고 추가 정보 설정 IP 주소에 해당하는 reverse DNS zone을 구성했다면, Add also reverse record 선택 → 자동으로 reverse DNS (PTR) zone에 추가됨 Screen Shot 2018-03-02 at 10.08.53 PM.png Step 3) 등록 결과 메시지 Screen Shot 2018-03-02 at 10.11.58 PM.png Step 4) 등록 결과 확인 (찾기) Screen Shot 2018-03-02 at 10.14.22 PM.png Troubleshooting Apple 장치에서 외부 주소 얻을 수 없는 경우 구성 후 Apple 장치 (Mac, iOS)에서 정상동작 하지 않을 수 있다. Apple 장치에서 외부 도메인 (예: google.com) 주소를 얻어오지 못하는 현상이 발생할 수 있다. 이 문제는 Recusor 구성을 통해 문제를 해소할 수 있다. Step 1) recursor 설치 $ yum install dns pdns-recursor Step 2) pdns 설정 변경 recursor 서비스 추가 (recursor=127.0.0.1:5678) launch+=gmysql # gmysql parameters gmysql-host=localhost gmysql-

2014 - 2023 CURVC Corp. All rights reserved

port= gmysql-dbname=powerdns gmysql-user=powerdns gmysql-password=curvc**** #gmysql-dnssec=yes # gmysql-socket= recursor=127.0.0.1:5678 Step 3) recursor 설정 forward-zones=i.curvc.com=127.0.0.1:5678;8.8.8.8 #forward-zones=<내부 전용 도메인, 외부와 공용이라면 '.'>=<내부 도메인 서버 주소>;<외부 도메인 서버 주소> local-port=5678 serve-rfc1918=no trace=on Step 4) 서비스 재시작 pdns와 pdns-recursor 서비스 재시작

2018-03-04 14:29:06.602

SELinux 환경에서 MySQL 데이터 폴더 변경하기

충분한 저장공간 또는 특별한 디스크에 MySQL 데이터 폴더를 위치시키는 경우가 많다. 이 때 폴더 접근 권한 설정은 SELinux를 고려해야한다. 이 페이지는 SELinux 에 대해 조금 소개하고 MySQL 데이터 폴더를 기본 위치에서 다른 경로로 이동하는 방법을 소개한다. Selinux 맛보기 Security-Enhanced Linux (SELinux)는 보안 강화를 위한 리눅스 커널 모듈로써 1998년 NSA와 RedHat이 제안하고 레드햇이 개발을 담당했다. 이 보안 모듈은 2003년 8월 8일 버전 2.6을 기점으로 주류 리눅스 커널에 통합되었고, CentOS 4에 처음 적용되었다. 해결하고 싶은 문제 SELinux는 아래의 문제 해결에 관심이 있다: 사용자 제어 불가: 사용자가 임의로 모든 유저자가 읽고 쓸 수 있는 폴더 또는 파일을 생성할 수 있음 프로세스가 보안 특성을 변경 가능: 사용자의 메일 파일은 해당 사용자만 읽을 수 있지만, 메일 client 프로세스가 해당 파일을 모든 사용자가 읽을 수 있도록 변경 가능 동작 모드 세 가지 동작 모드로 구성된다. Enforcing: SELinux 보안정책 사용을 강제하고 접근 위반 행위를 기록함 Permissive: SELinux가 사용되지만 보안정책을 강제하지는 않고 접근 위반에 대해 경고하고 위반 행위를 기록함 Disable: SELinux 사용 안함 SELinux 동작 상태 확인 방법: $ setstatus SELinux status: enabled SELinuxfs mount: /selinux Current mode: enforcing Mode from config file: enforcing Policy version: 21 Policy from config file: targeted 정책 SELinux는 기본적으로 접근을 거부하는 least-privilege 정책을 따른다. SELinux에서는 보안 정책을 "targeted policy"라는 용어로 정의하고 서로다른 보안정책간의 변환을 허용한다. 접근 제어 SELinux는 4가지 접근제어 형식 제공: Type Encorcement (TE): primary mechanism Role-Based Acess Control (RBAC): SELinux 사용자 기준의 접근제어 Multi-Level Security (MLS) Multi-Category Security (MCS): MLS의 확장으로 virtual machine과 container 분류 (compartment)에 사용됨 SELinux context 정보 확인 방법: $ ls -Z /var/lib/ drwxr-x--x. mysql mysql system_u:object_r:mysqld_db_t:s0 mysql $ ps axZ | grep mysqld system_u:system_r:mysqld_t:s0 1675 ? Sl 0:00 /usr/sbin/mysqld MySQL Data 폴더 변경 사전 조건 semanage 설치 $ yum install policycoreutils-python Step 1) MySQL 서비스 종료 $ systemctl stop mysqld Step 2) 데이터 폴더 이동 설정 변경을 최소화하기 위해 Linux의 symbolic link 사용 새로운 데이터 폴더 위치: /data/mysql-data $ mv /var/lib/mysql /data/mysql-data $ ln -s /data/mysql-data /var/lib/mysql Step 3) SELinux context 설정 $ semanage fcontext -a -t mysqld_db_t "/data/mysql-data(/.*)?" $ restorecon -Rv /data/mysql-data Step 4) MySQL 서비스 시작 $ systemctl start mysqld

2022-10-01 09:41:11.851

리눅스 루트 파일시스템 복구

루트 파일 시스템이 손상되어 부팅 자체가 안될 경우 파일시스템 복구를 시도하는 방법을 정리한다. 요약 복구 모드로 부팅 파티션 확인 복구 시도 파일 시스템 확인 재부팅 복구 복구 모드로 부팅 파일 시스템 오류가 발생하여 마운트에 실패하면 자동으로 복구 모드로 진입한다. 부팅이 가능한 상태라면, 부트 화면에서 "linux rescue" 를 입력한다. 파티션 확인 lvm 명령어로 진입하여 volume 정보를 확인한다. $ lvm (lvm) vgscan Screen Shot 2022-10-01 at 9.34.40 AM.png (lvm) quit 복구 시도 $ xfs_repair -L /dev/centos/root 만약, 파일 시스템 마운트가 가능하다면 '-L' 옵션을 제거한다. $ xfs_repair /dev/centos/root 파일 시스템 확인 마운트하여 파일시스템 내용을 확인한다. $ mkdir /tmp/mnt $ mount /dev/centos/root /tmp/mnt 재부팅 파워버튼을 눌러 시스템을 재기동한다.

2014 - 2023 CURVC Corp. All rights reserved

2019-02-16 08:29:37.235

Let's Encrypt 인증서 발급 방법

이 페이지는 Let's Encypt에서 운영하는 무료 SSL 인증서를 받는 방법에 대해 정리한다. 기준 OS CentOS 7 사전 준비 EPEL repository 설정 Certbot이 EPEL package이므로 EPEL repository를 사용할 수 있도록 설정한다 $ yum install epel-release 도메인 관리자 권한 SSL을 받을 때 _acme-challenge.<domain> 에 TXT record를 생성해야 하므로 도메인 관리자 계정이 필요하다. Certbot 설치 $ yum install python2-certbot-apache 인증서 받기 Certbot은 인증서를 받기 그리고 web server 설정에 반영하기를 지원한다. 본 정리에서는 인증서를 받는 방법에 대해 정리한다. Step 1) certbot 실행 $ certbot certonly -d <발급하려는 도메인 이름> -d <발급하려는 또 다른 도메인 이름> --manual --preferred-challenges dns-01 --force-renewal --manual-public-ip-logging-ok 발급하려는 도메인 이름 예) *.curvc.com *.almdemo.curvc.com jira.almdemo.curvc.com Step 2) _acme-challenge 레코드 생성 이 단계는 certbot (letsencrypt)가 도메인 소유주가 맞는지 확인하는 절차로써 certbot이 지정한 문자열을 지정된 도메인 (_acme-challenge.~) 레코드로 생성해 하고 certbot이 이를 확인하는 과정이다. - Please deploy a DNS TXT record under the name _acme-challenge.almdemo.curvc.com with the following value: ZFETHF7O9RWoct4tk8jkgrXIqjbZMBA0J1WLW5OITv4 Before continuing, verify the record is deployed. - Press Enter to Continue 도메인 서비스의 zone edit menu에서 _acme-challenge.<발급하려는 도메인 이름>에 TXT 형식의 레코드를 생성한다. 예) *.curvc.com 이라면: _acme-challenge.almdemo.curvc.com 에 TXT 값 (ZFETHF7O9RWoct4tk8jkgrXIqjbZMBA0J1WLW5OITv4) 레코드 생성 (기존에 존재하면 TXT 값 수정) 일반적으로 도메인 서버에 반영되기까지 시간이 필요하므로 충분히 (2분 이상) 기다리거나, 아래 방법으로 반영 여부를 확인후 Enter를 입력해 다음 단계로 진행한다. $ dig -t TXT _acme-challenge.almdemo.curvc.com ; <<>> DiG 9.10.6 <<>> -t TXT _acme-challenge.almdemo.curvc.com ;; global options: +cmd ;; Got answer: ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26540 ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0 ;; QUESTION SECTION: ;_acme-challenge.almdemo.curvc.com. IN TXT ;; ANSWER SECTION: _acme-challenge.almdemo.curvc.com. 301 IN TXT "RJULHF7O9RWoct4tk8jkgrXIqjbZMBA0J1WLW5OITv4" ;; Query time: 298 msec ;; SERVER: 10.0.0.3#53(10.0.0.3) ;; WHEN: Thu Sep 13 18:31:11 KST 2018 ;; MSG SIZE rcvd: 107 13 line과 같이 설정했던 값이 확인되면 다음 단계로 진행한다. Step3) 인증서 확인 다음 문구와 같이 발급된 인증서가 저장된 위치를 알려준다. 예) /etc/letsencrypt/live/almdemo.curvc.com IMPORTANT NOTES: - Congratulations! Your certificate and chain have been saved at: /etc/letsencrypt/live/almdemo.curvc.com/fullchain.pem Your key file has been saved at: /etc/letsencrypt/live/almdemo.curvc.com/privkey.pem Your cert will expire on 2018-12-12. To obtain a new or tweaked version of this certificate in the future, simply run certbot again. To non-interactively renew *all* of your certificates, run "certbot renew" Step 4) 인증서 사용 저장된 인증서를 필요한 곳에 사용하면 됨 인증서 갱신 $ certbot renew 참고 https://certbot.eff.org/lets-encrypt/centosrhel7-apache https://certbot.eff.org/lets-encrypt/centosrhel7-apache

2018-05-16 12:37:58.171

CentOS 6 Mac Terminal에서 Copy/paste 시 0~ 1~ 문자 추가되는 듄

환경 Mac의 terminal App을 이용해 ssh 프로토콜로 CentOS 6 접속 CentOS 6.9 Mac: High Sierra 현상 command를 복사하여 terminal에 붙여넣으면 0~ 1~ 추가됨 $ > 0~command 1~ 원인 Terminal의 bracket 붙여넣기 모드로 설정되어 있기 때문 해결 Terminal에 아래 명령 입력 printf "₩e[?2004l"

2020-11-01 09:27:51.028

복수의 Java 버전 운영하는 방법

2014 - 2023 CURVC Corp. All rights reserved

서버에 여러가지 Java 버전을 설치하는 경우가 자주 발생하는데, alternatives 도구를 이용해 간편하게 버전을 전환하는 방법을 정리한다. 적용 대상 CentOS, Red Hat Linux Ubuntu의 경우 alternative 대신 upeate-alternatives 명령어 사용 Java 설치 Java 설치 후 다음 절차를 통해 java, javac 등의 java 실행 파일 경로 등록 java 등록 $ sudo alternatives --install /usr/bin/java java <java 설치 폴더>/bin/java <우선 순위> 예) /usr/lib/jvm/open-jdk-11 에 설치한 경우 $ sudo alternatives --install /usr/bin/java java /usr/lib/jvm/open-jdk-11/bin/java 100 javac 등록 $ sudo alternatives --install /usr/bin/javac javac <java 설치 폴더>/bin/javac <우선 순위> 예) /usr/lib/jvm/open-jdk-11 에 설치한 경우 $ sudo alternatives --install /usr/bin/javac javac /usr/lib/jvm/open-jdk-11/bin/javac 100 jar 등록 $ sudo alternatives --install /usr/bin/jar jar <java 설치 폴더>/bin/jar <우선 순위> 예) /usr/lib/jvm/open-jdk-11 에 설치한 경우 $ sudo alternatives --install /usr /bin/jar jar /usr/lib/jvm/open-jdk-11/bin/jar 100 JAVA_HOME, JRE_HOME 등록 환경 설정 파일에 JAVA_HOME 환경변수를 지정한다. JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which java))))) export JAVA_HOME export CLASSPATH=.:$JAVA_HOME/jre /lib:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar Linux의 경우 JAVA_HOME, JRE_HOME 아래의 방법으로 환경설정이 가능하다. 본 문서는 profile.d 방식을 예로 설명한다. 시스템 수준 설정 /etc/profile /etc/profile.d/java.sh ← bash login shell에 적용됨 사용 수준 설정 ~/.bashrc ~/.bash_profile Java 버전 전환 설치된 java 버전 중 특정 버전을 선택할 수 있다. $ sudo alternatives --config java There are 3 programs which provide 'java'. Selection Command --- * 1 /usr/java/jdk1.8.0_151/jre/bin /java + 2 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/bin /java 3 /opt/jdk1.8.0_144/bin/java + : 사용자 지정 default, 수동 설정이 우선 * : 자동으로 설정된 default Alternatives 부가 기능 등록된 경로 제거 $ alternatives --remove <제거할 대상> 예) $ alternatives --remove /usr/lib/jvm /java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/bin/java

2022-03-30 05:55:55.651

리눅스 팁

이 페이지는 Linux와 관련된 팁을 정리한다.

2018-06-02 17:39:37.961

MySQL Resource Monitoring

2014 - 2023 CURVC Corp. All rights reserved

이 페이지는 모니터링 해야하는 MySQL metric에 대해 정리한다. Monitoring Metrics MySQL provides a few good metrics for monitoring your connections: Variable What it represents Why you should monitor it Threads_connected This variable indicates the total number of clients that have currently open connections to the server. It provides real-time information on how many clients are currently connected to the server. This can help in traffic analysis or in deciding the best time for a server re-start. Threads_running The number of threads that are not sleeping. It's good for isolating which connected threads are actively processing queries at any given time, as opposed to connections that are open but are currently idle. Connections The number of connection attempts (successful or not) to the MySQL server. It can give you a good idea of how many people and applications are accessing the database. Over time, these numbers reveal busiest times and average usage numbers. Connection_errors_internal The number of connections refused due to internal server errors, such as failure to start a new thread or an out-of-memory condition. Although MySQL exposes several metrics on connection errors, Connection_errors_internal is probably the most useful, because it is incremented only when the error comes from the server itself. Internal errors can indicate an out-of-memory condition or an inability to start a new thread. How to monitor We can use the MySQL show status command to show MySQL variables and status information. Here are a few examples: SHOW GLOBAL STATUS LIKE '% Threads_connected%'; +-------------------+-------+ | Variable_name | Value | +-------------------+-------+ | Threads_connected | 2 | +-------------------+-------+ SHOW GLOBAL STATUS LIKE '%Threads_running%'; +-----------------+-------+ | Variable_name | Value | +-----------------+-------+ | Threads_running | 1 | +-----------------+-------+ SHOW GLOBAL STATUS LIKE 'Connections'; +---------------+-------+ | Variable_name | Value | +---------------+-------+ | Connections | 20 | +---------------+-------+

2020-03-16 17:32:23.127

Apache Disabling TLS 1.0

개요 2018. 7. 30에 TLS 1.0 지원이 종료되었다. 이 페이지는 Apache webserver 에서 TLS 1.0 을 사용하지 않도록 설정하는 방법을 정리한다. /etc/httpd/conf.d/ssl. conf 변경 전 # SSL Engine Switch: # Enable/Disable SSL for this virtual host. SSLEngine on # List the protocol versions which clients are allowed to connect with. # Disable SSLv3 by default (cf. RFC 7525 3.1.1). TLSv1 (1.0) should be # disabled as quickly as practical. By the end of 2016, only the TLSv1.2 # protocol or later should remain in use. SSLProtocol all -SSLv3 SSLProxyProtocol all -SSLv3 $ nmap --script ssl-enum-ciphers -p 443 localhost | grep TLSv | TLSv1.0: | TLSv1.1: | TLSv1.2: 변경 후 # SSL Engine Switch: # Enable/Disable SSL for this virtual host. SSLEngine on # List the protocol versions which clients are allowed to connect with. # Disable SSLv3 by default (cf. RFC 7525 3.1.1). TLSv1 (1.0) should be # disabled as quickly as practical. By the end of 2016, only the TLSv1.2 # protocol or later should remain in use. SSLProtocol TLSv1.2 SSLProxyProtocoll TLSv1.2 $ nmap --script ssl-enum-ciphers -p 443 localhost | grep TLSv | TLSv1.2:

2018-03-29 17:19:05.593

VMWare 환경에서 CentOS 7 Root Partition Resizing

Created by Fredrik Mikker https://kb.op5.com/display/~fmikker, last modified by Martin Sharples https://kb.op5.com/display/~msharples on Jul 25, 2017 https://kb.op5.com/pages/diffpagesbyversion.action?pageId=5375707&selectedPageVersions=15&selectedPageVersions=16 Go to start of metadata https://kb.op5.com/display/COMMUNITY/Extend+root+partition+using+LVM#page-metadata-start Version This article was written for version 6 of OP5 Appliance System, it could work on both lower and higher version if nothing else is stated. Articles in community space is not covered by OP5 Support When running virtual machines there might be a need to add space your virtual machine when it's running low, in this example we will add 10GB to the root filesystem of a virtual instance of OP5 Appliance System 6.5. Prerequisites To follow this

2014 - 2023 CURVC Corp. All rights reserved

article the following will be needed: OP5 Appliance System 6 or CentOS 6 As always when working with storage, make sure you have a recent backup of the virtual machine to be safe if anything should go wrong. No snapshots can be present on the VM when resizing the disk Basic knowledge about Linux and LVM2 Shut down your virtual machine Change the setting "Provisioned Size" in VMware to the total size that you want to have available in the operating system of your virtual server: https://kb.op5.com /download/attachments/5375707/2016-11-15-001-lvm-provision-vdisk. png?version=1&modificationDate=1479224135000&api=v2 Start the virtual machine and log in as root to check that size on your disk has changed: # fdisk -l /dev/sda Disk /dev/sda: 107.4 GB, 107374182400 bytes 255 heads, 63 sectors/track, 13054 cylinders Units = cylinders of 16065 * 512 = 8225280 bytes Sector size (logical/physical): 512 bytes / 512 bytes I /O size (minimum/optimal): 512 bytes / 512 bytes Disk identifier: 0x000076f1 Device Boot Start End Blocks Id System /dev /sda1 * 1 17 131072 83 Linux Partition 1 does not end on cylinder boundary. /dev/sda2 17 8355 66976768 8e Linux LVM In this example, the disk /dev/sda has been extended to 107.4 GB and there are only two primary partitions currently in use, this is good because there are room for two more primary partitions. What you need to do now is to create the new partition sda3 that disk size can be taken from when extending the logical volume later on. This will be done with fdisk The default values here should be sane, just press enter after the command input. # fdisk /dev/sda WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u'). Command (m for help): n Command action e extended p primary partition (1-4) p Partition number (1-4): 3 First cylinder (8355-13054, default 8355): Using default value 8355 Last cylinder, +cylinders or +size{K,M,G} (8355-13054, default 13054): Using default value 13054 In the previous step, a new partition was created from the space that was added from VMware, and now we need to set the correct partition type before writing the partition table and saving the changes: Command (m for help): t Partition number (1-4): 3 Hex code (type L to list codes): 8e Changed system type of partition 3 to 8e (Linux LVM) Command (m for help): w The partition table has been altered! Calling ioctl() to re-read partition table. WARNING: Re-reading the partition table failed with error 16: Device or resource busy. The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8) Syncing disks. The warning message above can be addressed in two ways: 1) Reboot the system, 2) Re-scan the device and add the new partition. We will do the latter: # partx -v -a /dev/sda device /dev/sda: start 0 size 209715200 gpt: 0 slices dos: 4 slices # 1: 2048- 264191 (262144 sectors, 134 MB) # 2: 264192-134217727 (133953536 sectors, 68584 MB) # 3: 134217728- 209712509 (75494782 sectors, 38653 MB) # 4: 0- -1 (0 sectors, 0 MB) BLKPG: Device or resource busy error adding partition 1 BLKPG: Device or resource busy error adding partition 2 added partition 3 The partx application will give some errors for the existing partitions, but that's nothing to worry about. The important part is that partition 3 was successfully added. If partition 3 isn't added you need to reboot the machine to make the changes take effect. Now we need to create a new physical volume that can be used for adding space to the volume group # pvcreate /dev/sda3 Physical volume "/dev/sda3" successfully created Check what name your volume group your root filesystem uses: # vgdisplay --- Volume group --- VG Name 1 System ID Format lvm2 [..] In this case the Volume group is called 1 Now, lets extend the volume group with the space we added with vgextend: # vgextend 1 /dev/sda3 Volume group "1" successfully extended With the utility pvscan you can see what physical volumes are available, and how much free space there is within them. The free space can be used to extend your root partion, or other volumes that are short on space. # pvscan PV /dev/sda2 VG 1 lvm2 [63,87 GiB / 29,87 GiB free] PV /dev/sda3 VG 1 lvm2 [36,00 GiB / 36,00 GiB free] Total: 2 [99,87 GiB] / in use: 2 [99,87 GiB] / in no VG: 0 [0] Now in this step we will at 10G to the root partition. First find out what logical volume that the root partition belongs to: # df - h Filesystem Size Used Avail Use% Mounted on /dev/mapper/1- sys 7,8G 1,4G 6,0G 20% / tmpfs 246M 0 246M 0% /dev /shm /dev/sda1 120M 27M 88M 24% /boot /dev/mapper/1- opt 7,8G 181M 7,2G 3% /opt /dev/mapper/1-tmp 7,8G 19M 7,4 G 1% /tmp /dev/mapper/1-var 7,8G 475M 6,9G 7% /var In this case, the root partition originates from the logical volume "sys", and the total size is 7.8 GB. The following command will add 10 GB to the Logical Volume "sys" in the Volume Group "1" and the disk area will be taken from the Physical Volume sda3. # lvextend -L+10G /dev/mapper/1-sys /dev /sda3 Size of logical volume 1/sys changed from 8,00 GiB (2048 extents) to 18,00 GiB (4608 extents). Logical volume sys successfully resized. Run the following command to expand the ext4 filesystem inside of the Logical Volume: # xfs_growfs -d /dev/mapper/cl-root meta-data=/dev/mapper/cl- root isize=512 agcount=4, agsize=2293504 blks = sectsz=512 attr=2, projid32bit=1 = crc=1 finobt=0 spinodes=0 data = bsize=4096 blocks=9174016, imaxpct=25 = sunit=0 swidth=0 blks

2014 - 2023 CURVC Corp. All rights reserved

naming =version 2 bsize=4096 ascii-ci=0 ftype=1
log =internal bsize=4096 blocks=4479,
version=2 = sectsz=512 sunit=0 blks, lazy-count=1
realtime =none extsz=4096 blocks=0, rtextents=0 data blocks
changed from 9174016 to 28865536 And voli! Now your root partition is
extended with 10 GB: # df -h Filesystem Size Used Avail Use%
Mounted on /dev/mapper/1-sys 18G 1,5G 16G 9% /
tmpfs 246M 0 246M 0% /dev/shm /dev
/sda1 120M 27M 88M 24% /boot /dev/mapper/1-opt 7,8
G 181M 7,2G 3% /opt /dev/mapper/1-tmp 7,8G 19M 7,4G 1% /tmp
/dev/mapper/1-var 7,8G 475M 6,9G 7% /var

2022-03-30 05:58:01.898

쓰래드 개수 확인하기

특정 프로세스의 thread 개수 확인하는 방법 ps -o thcount <pid> 지속적으로 개수 모니터링 하는 방법 watch 명령은 주어진 명령어를 주기적으로 실행하여 출력을 화면에 표시한다. default 주기: 2초 종료 방법: Ctrl + C watch ps -o thcount <pid>

2022-09-22 22:51:51.831

Artifactory 설치 가이드

2014 - 2023 CURVC Corp. All rights reserved

이 문서는 Artifactory 설치 가이드를 공유한다. 설치 환경 OS : CentOS 7 Java : jdk1.8 Database : Postgresql 9.3 Artifactory Install Artifactory를 위한 Database & DBuser 생성 $ su - postgres -bash$ psql postgres=# CREATE USER artifactory with PASSWORD 'artifactory'; postgres=# CREATE DATABASE artifactory WITH OWNER=artifactory ENCODING='UTF8'; postgres=# GRANT ALL PRIVILEGES ON DATABASE artifactory TO artifactory; postgres=# ₩q -bash$ exit $ Artifactory 다운로드 Artifactory를 다운로드 받는다.(https://bintray.com/jfrog/artifactory/jfrog-artifactory-oss-zip/view https://bintray.com/jfrog/artifactory/jfrog-artifactory-oss-zip/view) Pro version: https://releases.jfrog.io/ui/ https://releases.jfrog.io/ui/ 압축을 해제한 후 /opt/artifactory로 이동 시킨다. $ ls /opt/artifactory artifactory-oss-5.1.0.zip Artifactory port 변경 기본 포트는 8081이다. 충돌이 날 경우 다른 포트로 변경한다. $ vi /opt/artifactory/tomcat/conf/server.xml $ cat /opt/artifactory/tomcat/conf/server.xml <Server port="8015" shutdown="SHUTDOWN"> <Service name="Catalina"> <Connector port="8081"/> <!-- This is the optional AJP connector --> <Connector port="8019" protocol="AJP/1.3"/> <Engine name="Catalina" defaultHost="localhost"> <Host name="localhost" appBase="webapps"/> </Engine> </Service> <Server> Artifactory 시작 /opt/artifactory/bin/artifactory.sh start 명령어로 실행 12.install.png Artifactory web 접속 http://ipaddress:8081 http://ipaddress:8081 접속 한다. 기본 port는 8081이다. Next 버튼을 클릭 한다. 4.install.png 패스워드 입력 후 Next 버튼을 클릭 한다. 5.install.png Configure a Proxy Server Skip 버튼을 클릭한다. 6.install.png Create Repositories 필자는 일단 skip을 하였다. 7.install.png Finish 버튼을 클릭 하여 마친다. 8.install.png Admin > Advanced > System Info 클릭하여 정보를 확인 한다. 9.install.png Database가 derby 로 설치가 완료 되었다. 13.install.png Artifactory Database 변경 Artifactory를 중단 artifactory database를 변경하기위해 artifactory를 중단 한다. 14.install.png Artifactory Database 설정 변경 기존 db.properties를 파일을 old로 이름을 변경하고 postgresql.properties를 복사하여 db.properties으로 변경 한다. vi 편집기로 db.properties를 정보를 Database Postgresql 설정에 맞게 변경 하고 저장 한다. 15.install.png Postgresql Jdbc driver 다운로드 Artifactory 실행시 postgresql jdbc driver가 없어 Error가 발생한다. 따라서 postgresql jdbc driver를 다운 받는다. cd /opt/artifactory/tomcat/lib wget https://jdbc.postgresql.org/download/postgresql-9.3-1104.jdbc41.jar https://jdbc.postgresql.org/download/postgresql-9.3-1104.jdbc41.jar 16.install.png Artifactory 시작 Artifactory web 접속을 하기 위해 실행 명령어를 실행한다. 17.install.png Artifactory web 접속 http://ipaddress:8081 http://ipaddress:8081/ 접속 한다. 기본 port는 8081이다. Next 버튼을 클릭 한다. 4.install.png 패스워드 입력 후 Next 버튼을 클릭 한다. 5.install.png Configure a Proxy Server Skip 버튼을 클릭한다. 6.install.png Create Repositories 필자는 일단 skip을 하였다. 7.install.png Finish 버튼을 클릭 하여 마친다. 8.install.png Admin > Advanced > System Info 클릭하여 정보를 확인 한다. 9.install.png Database가 Postgresql으로 변경 되었음을 확인 할 수 있다. 21.install.png

2020-10-16 09:43:22.593

Nginx 설치하기(인터넷 환경)

이 문서는 CentOS 7에 Nginx 설치 가이드를 공유하기 위해 작성되었다. 페이지 속성은 아래와 같이 유지합니다. 제품 및 버전 Nginx DB 및 버전 MySQL 5.7.32 JAVA 버전 Oracle JDK 1.8.0.161 OS 버전 CentOS 7 목차 Nginx 설치 순서 1. Nginx 저장소 추가 # vim /etc/yum.repos.d/nginx.repo 2. 저장소에 아래 내용 추가 [nginx] name=nginx repo baseurl=http://nginx.org/packages/centos/7/$basearch/ gpgcheck=0 enabled=1 3. Nginx 설치 # yum install -y nginx 4. 방화벽 웹포트 허용 # firewall-cmd -permanent -zone=public -add-service=http # firewall-cmd -permanent -zone=public -add-service=https # firewall-cmd -reload 5. Nginx 서비스 등록 # systemctl start nginx # systemctl enable nginx 6. Nginx 정상 설치 확인 웹브라우저에서 http://설치한ip주소 입력시 아래와 같이 노출되면 정상적으로 설치된것 캡처.PNG 코드는 한줄이라도 코드 블록을 사용합니다. 이미지 사이즈는 700px 이하로 설정합니다. 가급적이면 테두리를 사용하지 않습니다.

2018-03-09 20:36:40.02

Ldap 설치 가이드

2014 - 2023 CURVC Corp. All rights reserved

이 문서는 LDAP 설치 가이드를 공유하기 위해 작성되었다. Ubuntu Install slapd 설치를 합니다. sudo apt-get install slapd slapd 관리자 패스워드를 입력 합니다. 2.install.png 다시한번 패스워드를 입력 합니다. 3.install.png slapd 설정을 재 설정 합니다. sudo dpkg-reconfigure slapd Yes를 선택하면 ldap 설정을 생략하기 때문에 No를 선택 합니다. 4.install.png DNS 도메인을 입력합니다.(ex : DN주소 dc=curvc,dc=com) 5.install.png Base DN의 이름을 입력합니다.(ex: 회사 or 부서) 6.install.png slapd 관리자 패스워드를 입력합니다. 7.install.png 다시 한번 패스워드를 입력합니다. 8.install.png Database는 어떤걸로 설정 할지 정합니다.(기본 추천은 HDB 입니다.) 9.install.png slapd를 삭제 할때 Database 정보까지 삭제를 안 하도록 선택 합니다. 10.install.png 기존 설정들을 이동 하도록 선택 합니다. 11.install.png LDAPv3 Protocol 만 들어오도록 설정을 안 합니다. 12.install.png CentOS 7 Install openldap을 yum 으로 설치 합니다. yum install -y openldap* slapd를 서비스를 시작 합니다. systemctl start slapd.service systemctl enable slapd.service slapd 패스워드 등록 slappasswd New password: Re-enter new password: {SSHA}fvEBG14rJvKy+ROY2tQJsEkCFf2VNxnU db.ldif 파일 생성 후 아래 코드 입력 후 Ldap에 import 한다. vi db.ldif dn: olcDatabase={0}config, cn=config changetype: modify replace: olcRootPW olcRootPW: {SSHA} fvEBG14rJvKy+ROY2tQJsEkCFf2VNxnU :wq! ldapmodify -Y EXTERNAL -H ldapi:/// -f db.ldif Ldap 스키마를 추가 한다. ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif ldapadd -Y EXTERNAL -H ldapi:/// -f /etc /openldap/schema/nis.ldif ldapadd -Y EXTERNAL -H ldapi:/// -f /etc /openldap/schema/inetorgperson.ldif domain.ldif 파일 생성 후 아래 코드 입력 후 Ldap에 Import 한다. vi domain.ldif dn: olcDatabase={2}hdb,cn=config changetype: modify replace: olcSuffix olcSuffix: dc=curvc,dc=com dn: olcDatabase={2}hdb,cn=config changetype: modify replace: olcRootDN olcRootDN: cn=admin,dc=curvc,dc=com dn: olcDatabase={2}hdb,cn=config changetype: modify add: olcRootPW olcRootPW: {SSHA} fvEBG14rJvKy+ROY2tQJsEkCFf2VNxnU dn: olcDatabase={2}hdb,cn=config changetype: modify add: olcAccess olcAccess: {0}to attrs=userPassword, shadowLastChange by dn="cn=admin,dc=curvc,dc=com" write by anonymous auth by self write by * none olcAccess: {1}to dn.base="" by * read olcAccess: {2}to * by dn="cn=admin,dc=curvc,dc=com" write by * read :wq! ldapmodify -Y EXTERNAL -H ldapi:/// -f domain.ldif basedomain 파일 생성 후 아래 코드 입력 후 Ldap에 import 한다. Import시 패스워드를 묻는다 초기 slappasswd 를 입력 한다. vi basedomain.ldif dn: dc=curvc,dc=com objectClass: top objectClass: dcObject objectclass: organization o: Server World dc: curvc dn: cn=admin,dc=curvc,dc=com objectClass: organizationalRole cn: admin description: Directory Manager dn: ou=People, dc=curvc,dc=com objectClass: organizationalUnit ou: People dn: ou=Group, dc=curvc,dc=com objectClass: organizationalUnit ou: Group :wq! ldapadd -x -D cn=admin,dc=curvc,dc=com -W -f basedomain.ldif LDAP Admin Tool를 이용해서 접속해 본다.(http://www.ldapadmin.org/download/ldapadmin.html http://www.ldapadmin.org/download/ldapadmin.html) 1.install.png LDAP 구성이 된 것을 확인 할 수 있다. 2.install.png

2018-03-09 20:37:27.574

SSH 설치 가이드

이 문서는 SSH 설치에 대한 정보를 공유하기 위해 작성되었다. Cent OS ssh 설치 ssh 설치를 위해 다음 명령을 실행할 수 있다. yum install openssh-server openssh-clients openssh-askpass ssh server 포트 변경 ssh server의 설정을 위해 다음 파일을 편집한다. vi /etc/sshd_config Port 22 PermitRootLogin no ssh 실행 ssh 서버를 실행하기 위해 다음 명령을 수행한다. /etc/init.d/sshd start ssh 구동 확인 ssh 구동 확인을 위해 다음을 수행합니다. ps -ef | grep sshd

2014 - 2023 CURVC Corp. All rights reserved