□□□ को कॉन्फ़िगर करना

मुख्य बिंदु

- □ □□□ को कॉन्फ़िगर करने में .npmrc फ़ाइल को संपादित करने की संभावना है, जिसमें प्रॉक्सी और रजिस्ट्री सेटिंग्स शामिल हैं, जो मैन्युअल या कमांड्स के माध्यम से किया जा सकता है।
- □ अनुसंधान का सुझाव है कि प्रॉक्सी सेटिंग्स को proxy = http://user:password@proxy_server:port प्रारूप में रखना चाहिए, जिसमें विशेष चरित्र □□□ एन्कोडेड हों।
- □ सबूत इस ओर इशारा करते हैं कि रजिस्ट्री को registry = https://registry.npmjs.org/ के रूप में डिफ़ॉल्ट में सेट किया जाना चाहिए, जिसमें □□□ वरिफिकेशन को निष्क्रिय करने की विकल्प उपलब्ध हैं।

□□□ और .npmrc के साथ कॉन्फ़िगर करने

□□□, या नोड पैकेज मैनेजर, जावास्क्रिप्ट डिपेंडेंसियों को मैनेज करने का एक टूल है, और इसे कॉन्फ़िगर करने में आम तौर पर रजिस्ट्री और प्रॉक्सी के लिए .npmrc फ़ाइल शामिल होती है।

.npmrc क्या है और यह कहाँ स्थित है? .npmrc फ़ाइल □□□ का एक कॉन्फ़िगरेशन फ़ाइल है, जो इसे कैसे व्यवहार करे, इस पर कस्टमाइज़ेशन की अनुमति देता है। यह हो सकता है: - **ग्लोबल**: यूनिक्स-आधारित सिस्टम पर ~./.npmrc पर पाया जाता है या विंडोज पर %APPDATA%\npm\npmrc (आम तौर पर C:\Users\<username>\AppData\Roaming\npm\npmrc). - **प्रोजेक्ट के लिए स्थानीय**: प्रोजेक्ट के रूट डायरेक्टरी में ./package/.npmrc पर स्थित है। स्थानीय सेटिंग्स ग्लोबल सेटिंग्स को ओवरराइड करते हैं।

प्रॉक्सी और □□□□□ प्रॉक्सी सेटिंग प्रॉक्सी के पीछे काम करने के लिए, .npmrc में ये लाइनें जोड़ें: - proxy = http://user:password@proxy_server:p
- https_proxy = http://user:password@proxy_server:port

ध्यान दें कि https_proxy के लिए भी, प्रोटोकॉल के रूप में http:// का उपयोग करें। अगर आपका यूजरनेम या पासवर्ड में विशेष चरित्र हैं (जैसे, \, @), तो उन्हें □□□ एन्कोड करें (जैसे, domain\user को domain%5Cuser में बदलें)। आप इनको कमांड्स के माध्यम से भी सेट कर सकते हैं: - npm config set proxy http://user:password@proxy_server:port - npm config set https_proxy http://user:password@proxy_server:port

रजिस्ट्री सेटिंग रजिस्ट्री वह □□□ है जिसे □□□ पैकेजों को फेटच करने के लिए उपयोग करता है, आम तौर पर सेट किया जाता है: - registry = https://registry.npmjs.org/

कस्टम रजिस्ट्रियों के लिए, अपने □□□ से बदलें (जैसे, किसी कंपनी की निजी रजिस्ट्री)। इसे सेट करने के लिए: - npm config set registry https://your_custom_registry.com/

□□□ और वरिफिकेशन का समाधान अगर आप □□□ समस्याओं का सामना करते हैं, तो विचार करें: - .npmrc में strict-ssl = false को सेट करें ताकि □□□ त्रुटियों को नजरअंदाज कर सकें, हालांकि यह कम सुरक्षित है। इसे सेट करने के लिए: - npm config set strict-ssl false

कॉन्फ़िगरेशन का उपयोग और वरिफिकेशन .npmrc को किसी भी टेक्स्ट एडिटर के साथ सीधे संपादित करें, या npm config set कमांड्स का उपयोग करें। वरिफिकेशन के लिए: - सभी सेटिंग्स को सूचीबद्ध करें npm config list के साथ। - एक विशिष्ट सेटिंग को npm config get proxy के साथ चेक करें। - समस्याओं को npm install --verbose के साथ डिबग करें, जो विस्तृत आउटपुट के लिए है।

एक उदाहरण .npmrc फ़ाइल इस तरह दिख सकती है:

```
proxy = http://user:password@proxy_server:8080
https_proxy = http://user:password@proxy_server:8080
registry = https://registry.npmjs.org/
strict-ssl = false
```

सर्वेक्षण नोट: □□□ को .npmrc के साथ कॉन्फ़िगर करने का व्यापक गाइड

इस खंड में □□□ को कॉन्फ़िगर करने का विस्तृत अन्वेषण शामिल है, जो रजिस्ट्री और प्रॉक्सी सेटिंग्स के लिए .npmrc फ़ाइल पर केंद्रित है, जिसमें सीधे उत्तर के साथ अतिरिक्त संदर्भ और तकनीकी विवरण शामिल हैं।

.npmrc का परिचय और इसका भूमिका .npmrc फ़ाइल □□□ का एक महत्वपूर्ण कॉन्फ़िगरेशन फ़ाइल है, जो पैकेज मैनेजमेंट व्यवहार को कस्टमाइज़ करने की अनुमति देता है। यह एक □□□ प्रारूप का समर्थन करता है, जिसमें कुंजी-मान्यता जोड़े होते हैं, जहां टिप्पणियाँ एक सेमिकॉलन (;) से शुरू होती हैं। यह ग्लोबल या स्थानीय रूप में स्थित हो सकता है: - **ग्लोबल स्थान**: यूनिक्स-आधारित सिस्टम पर, यह ~./.npmrc पर है; विंडोज पर, यह %APPDATA%\npm\npmrc पर है, आम तौर पर C:\Users\<username>\AppData\Roaming\npm\npmrc. - **स्थानीय स्थान**: प्रोजेक्ट के रूट डायरेक्टरी में ./package/.npmrc के रूप में। स्थानीय सेटिंग्स ग्लोबल सेटिंग्स को ओवरराइड करते हैं, जो प्रोजेक्ट-खास कॉन्फ़िगरेशन प्रदान करते हैं।

इस हिरार्की के कारण फ्लेक्सिबिलिटी होती है, जो डेवलपर्स को कई प्रोजेक्टों या साझा सिस्टमों में सेटिंग्स को दोनों यूजर और प्रोजेक्ट स्तर पर मैनेज करने की अनुमति देता है।

विस्तृत प्रॉक्सी कॉन्फ़िगरेशन प्रॉक्सी के पीछे काम करने के लिए, .npmrc में दोनों proxy और https_proxy सेट करने की आवश्यकता होती है। प्रारूप है: -proxy = http://user:password@proxy_server:port -https_proxy = http://user:password@proxy_server:port

ध्यान दें कि https_proxy प्रोटोकॉल के रूप में http:// का उपयोग करता है, नहीं https://, क्योंकि प्रॉक्सी सर्वर कनेक्शन को हैंडल करता है, खुद को प्रॉक्सी तक एन्क्रिप्ट नहीं करता है। यह एक आम बिंदु है, क्योंकि प्रॉक्सी तक के पेलोड को डेस्टिनेशन तक □□□-एन्क्रप्ट किया जाता है।

प्रमाणिकरण के लिए, यूजरनेम और पासवर्ड शामिल करें, लेकिन अगर वे विशेष चरित्रों को शामिल करते हैं, तो □□□ एन्कोडिंग आवश्यक है: - बैकस्लैश (\) को %5C में बदलें। - एट सिम्बल (@) को %40 में बदलें। - कोलन (:) को %3A में बदलें।

उदाहरण के लिए, अगर यूजरनेम domain\user है और पासवर्ड pass@word है, तो यह होगा proxy = http://domain%5Cuser:pass%40word@proxy_se
इस एन्कोडिंग से □□□ द्वारा सही तरह से पर्स किया जाता है।

विंडोज उपयोगकर्ता, विशेष रूप से एक्टिव डायरेक्टरी के साथ कॉर्पोरेट वातावरण में, बैकस्लैश के साथ समस्याओं का सामना कर सकते हैं। कुछ रिपोर्टों का सुझाव है कि .npmrc में domain\username को पढ़ने पर domain/username में बदल सकता है, जो प्रमाणिकरण विफलता का कारण बन सकता है। ऐसे मामलों में, □□□□ □□□□□□□□□□□□□□ □□□□□□ □□□□□□□ या □□□□□□ जैसे टूल्स का उपयोग कर सकते हैं, हालांकि वे सीधे .npmrc कॉन्फ़िगरेशन के बाहर हैं।

रजिस्ट्री कॉन्फ़िगरेशन और □□□ विचार रजिस्ट्री सेटिंग यह □□□ है जिसे □□□ पैकेजों को फेटच करने के लिए उपयोग करता है, डिफ़ॉल्ट में सेट किया जाता है: - `registry = https://registry.npmjs.org/`. कस्टम या निजी रजिस्ट्रियों के लिए, सेट करें: - `registry = https://your_custom_registry.com/`

अगर □□□ वरिफिकेशन विफल हो जाता है, आम तौर पर कॉर्पोरेट प्रॉक्सी के साथ स्व-साइन कर्टिफिकेट के कारण, तो आप स्ट्रिक्ट □□□ को निष्क्रिय कर सकते हैं: - `strict-ssl = false`

हालांकि, यह सुरक्षा को कम करता है, क्योंकि यह सर्टिफिकेट वरिफिकेशन को बायपास करता है। इसके बजाय, सर्टिफिकेट प्राधिकरण को कॉन्फ़िगर करें: - `npm config -g set cafile [YOUR_CERTIFICATE_DIR]/[CERTIFICATE_NAME].crt` का उपयोग करें ताकि एक ट्रस्टेड □□ फ़ाइल को स्पेसिफाई करें।

कुछ डेवलपर्स `registry = http://registry.npmjs.org/` को सेट करने का विकल्प चुनते हैं ताकि □□□ को पूरी तरह से टाल सकें, लेकिन यह सुरक्षा के लिए अनुशंसित नहीं है, विशेष रूप से प्रोडक्शन में।

`.npmrc` का उपयोग: मैन्युअल संपादन □□. कमांड लाइन आप किसी भी टेक्स्ट एडिटर के साथ सीधे `.npmrc` को संपादित कर सकते हैं, सुनिश्चित करते हुए कि प्रत्येक सेटिंग एक नए लाइन पर है, प्रारूप `key = value` के साथ। टिप्पणियाँ `;` से शुरू होती हैं, उदाहरण के लिए:

```
;
proxy = http://user:password@proxy_server:8080
https_proxy = http://user:password@proxy_server:8080
registry = https://registry.npmjs.org/
strict-ssl = false
```

अल्टर्नेटिव रूप से, सुविधा के लिए □□□ कमांड्स का उपयोग करें: - प्रॉक्सी सेट करें: `npm config set proxy http://user:password@proxy_server:p` - रजिस्ट्री सेट करें: `npm config set registry https://registry.npmjs.org/` - स्ट्रिक्ट-□□□ सेट करें: `npm config set strict-ssl false`

ये कमांड्स सही `.npmrc` फ़ाइल को अपडेट करते हैं, जो संदर्भ (ग्लोबल या स्थानीय) के आधार पर है। ग्लोबल और स्थानीय के बीच स्विच करने के लिए, ग्लोबल के लिए `-g` का उपयोग करें, उदाहरण के लिए, `npm config set -g proxy ....`

वरिफिकेशन और ट्रबलशूटिंग कोनफ़िगरेशन को वरिफिकेशन के लिए, उपयोग करें: - `npm config list` सभी सेटिंग्स को सूचीबद्ध करने के लिए, जो दोनों प्रभावी और डिफ़ॉल्ट मान्यताओं को दिखाता है। - `npm config get proxy` एक विशिष्ट सेटिंग को प्राप्त करने के लिए।

प्रॉक्सी के पीछे, विशेष रूप से, समस्याओं को हल करने के लिए, चलाएं: - `npm install --verbose` विस्तृत लॉग्स को देखने के लिए, जो कनेक्शन समस्याओं, जैसे `ECONNREFUSED` या `socket hang up`, को पहचानने में मदद कर सकते हैं।

अगर सेटिंग्स प्रभावी नहीं होतीं, तो पर्यावरण चरित्रों (`HTTP_PROXY`, `HTTPS_PROXY`) को चेक करें जो `.npmrc` सेटिंग्स को ओवरराइड कर सकते हैं। ये सिस्टम-वाइड सेट हो सकते हैं और प्राथमिकता रखते हैं, इसलिए सुसंगतता सुनिश्चित करें।

बेस्ट प्रैक्टिस और अतिरिक्त विचार

- **सुरक्षा**: प्रोडक्शन में `strict-ssl = false` से बचें; बजाय इसके, ट्रस्टेड सर्टिफिकेट्स को कॉन्फ़िगर करें।
- **पर्यावरण चरित्र**: `NODE_ENV` और `HTTPS_PROXY` को ध्यान में रखें जो `.npmrc` सेटिंग्स को ओवरराइड कर सकते हैं, जो □□/□□ पाइपलाइंस के लिए उपयोगी हो सकते हैं, लेकिन स्थानीय रूप से भ्रमित हो सकते हैं।

- **लॉग लेवल**: `.npmrc` में `loglevel` को (जैसे, `loglevel = http`) डिबगिंग के दौरान विस्तृत आउटपुट के लिए सेट करें, जिसमें विकल्प `silent` से `silly` (7 लेवल्स) तक हैं।
- **कॉर्पोरेट प्रॉक्सी**: □□□□ प्रॉक्सी के लिए, □□□□□ जैसे टूल्स का विचार करें, एक स्थानीय प्रॉक्सी सर्वर सेटअप करें, और `.npmrc` को `localhost:3128` पर सेट करें, जिससे प्रमाणिकरण सरल हो जाता है।

उदाहरण कॉन्फ़िगरेशन यहाँ एक टेबल है जो प्रॉक्सी और रजिस्ट्री के लिए आम `.npmrc` सेटिंग्स को सारांशित करता है, साथ ही उदाहरण के साथ:

सेटिंग	विवरण	उदाहरण
□□□□□	□□□□ प्रॉक्सी सर्वर □□□	`proxy = http://user:password@proxy:8080`
□□□□□_□□□□□	□□□□□ प्रॉक्सी सर्वर □□□	`https_proxy = http://user:password@proxy:8080`
□□□□□□□□	पैकेज रजिस्ट्री □□□	`registry = https://registry.npmjs.org/`
□□□□□□-□□□	□□□ वरिफिकेशन को सक्षम/असक्षम करना	`strict-ssl = false`
□□□□□□□□	कंसोल आउटपुट लेवल (□□□□□□ से □□□□□, 7 लेवल्स)	`loglevel = http`

इस टेबल से नए डेवलपर्स के लिए जल्दी संदर्भ के लिए मदद मिलती है, विशेष रूप से `.npmrc` के साथ नए हैं।

अप्रत्याशित विवरण: ग्लोबल □□. स्थानीय सेटिंग्स पर प्रभाव एक अप्रत्याशित पहलू यह है कि ग्लोबल `.npmrc` सेटिंग्स सभी प्रोजेक्टों को प्रभावित करती हैं, जबकि स्थानीय सेटिंग्स विशिष्ट प्रोजेक्टों के लिए ओवरराइड कर सकती हैं। इस लेअरिंग से भ्रम हो सकता है, अगर, उदाहरण के लिए, एक ग्लोबल प्रॉक्सी सेट है लेकिन एक स्थानीय प्रोजेक्ट एक अलग रजिस्ट्री का उपयोग करता है, जो प्रमाणिकरण समस्याओं का कारण बन सकता है। इस हिरार्की को समझना बहु-प्रोजेक्ट वातावरणों के लिए महत्वपूर्ण है।

निष्कर्ष प्रॉक्सी और रजिस्ट्री सेटिंग्स के लिए □□□ को `.npmrc` के साथ कॉन्फ़िगर करने में सही प्रारूप और एन्कोडिंग और □□□ के विचार के साथ सरल है। सीधे संपादित करके या कमांड्स का उपयोग करके, और `npm config list` के साथ वरिफिकेशन करके, आप सुनिश्चित कर सकते हैं कि पैकेज मैनेजमेंट, विशेष रूप से कॉर्पोरेट नेटवर्क के साथ प्रॉक्सी के साथ, सुलभ हो जाए। और अधिक पढ़ने के लिए, आधिकारिक दस्तावेज़ देखें □□□ □□□ □□□□□□ या बेस्ट प्रैक्टिस देखें □□□□□□□□□□□ □□□□ .□□□□□.