Branching Workflows - - -  
अब तपाईंले शाखा बनाउने र मर्ज गर्ने आधारभूत कुरा बुझिसकेपछि, तपाईंले तीसँग के गर्न सक्नुहुन्छ वा गर्नुपर्छ? यस खण्डमा, हामी केही साधारण workflow हरूलाई कभर गर्नेछौं जुन यो हल्का शाखाकरणले सम्भव बनाउँछ, ताकि तपाईं यसलाई आफ्नो development cycle मा समावेश गर्न चाहनुहुन्छ कि छैन निर्णय गर्न सक्नुहुन्छ।

Long-Running Branches - - -  
किनभने Git ले साधारण three-way merge प्रयोग गर्छ, लामो समयसम्म एउटै शाखाबाट अर्को शाखामा पटक-पटक मर्ज गर्नु सामान्यतया सजिलो हुन्छ। यसको मतलब तपाईंसँग विभिन्न development cycle का चरणहरूको लागि सधैं खुला रहने धेरै शाखाहरू हुन सक्छन्, जसमा तपाईं नियमित रूपमा केही शाखाबाट अरू शाखामा मर्ज गर्न सक्नुहुन्छ।

धेरैजसो Git developer हरूसँग यस्तो workflow हुन्छ जसले यस दृष्टिकोणलाई अपनाउँछ। जस्तै, आफ्नो master शाखामा मात्र पूर्ण रूपमा स्थिर code राख्नु — सम्भवतः मात्र त्यस्तो code जुन जारी भइसकेको छ वा हुनेवाला छ। उनीहरूसँग develop वा next नामको अर्को parallel शाखा हुन्छ जसबाट काम गरिन्छ वा स्थिरता जाँच्न प्रयोग गरिन्छ। यो सधैं स्थिर नहुन सक्छ, तर जब यो स्थिर अवस्थामा पुग्छ, यसलाई master मा मर्ज गर्न सकिन्छ।

यस शाखालाई प्रायः topic branches (छोटो अवधिको शाखाहरू, जस्तै तपाईंको पहिलेको iss53 शाखा) तान्नका लागि प्रयोग गरिन्छ, जब तिनीहरू तयार हुन्छन्, सुनिश्चित गर्नका लागि कि तिनीहरूले सबै परीक्षण पास गर्छन् र कुनै नयाँ bug ल्याउँदैनन्।

वास्तवमा, हामी commit history मा pointers लाई माथि र तल सार्ने कुरा गर्दैछौं। स्थिर शाखाहरू commit history मा तलतिर हुन्छन्, र bleeding-edge शाखाहरू history मा माथितिर।

A diagram of a company

Description automatically generated

यो सामान्यतया कामको silo जस्तै सोच्न सजिलो हुन्छ, जहाँ commit को समूहहरू पूर्ण रूपमा परीक्षण भएपछि अधिक स्थिर silo मा उक्लिन्छन्।

A diagram of a process

Description automatically generated with medium confidence

तपाईं यो प्रक्रिया स्थिरताको धेरै तहहरूको लागि जारी राख्न सक्नुहुन्छ। केही ठूला प्रोजेक्टहरूसँग **proposed** वा **pu (proposed updates)** शाखा पनि हुन्छ, जसमा यस्ता शाखाहरू समाहित हुन्छन् जुन **develop** वा **master** शाखामा जान तयार छैनन्।  
यसको मुख्य विचार यो हो कि तपाईंको शाखाहरू विभिन्न तहहरूको स्थिरतामा हुन्छन्; जब तिनीहरू अधिक स्थिर तहमा पुग्छन्, तिनीहरू माथिल्लो शाखामा मर्ज गरिन्छन्।

धेरैजसो अवस्थामा धेरै **long-running** शाखाहरू आवश्यक छैनन्, तर यो ठूला वा जटिल प्रोजेक्टहरूसँग काम गर्दा प्रायः उपयोगी हुन्छ।

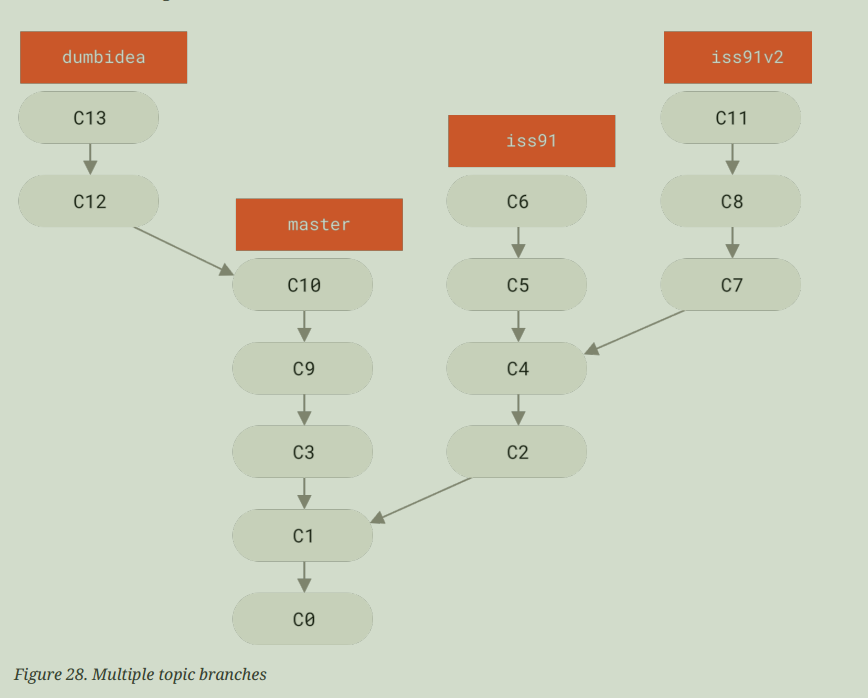
Topic Branches - - - -

**टपिक ब्रान्चहरू** कुनै पनि आकारका प्रोजेक्टहरूमा उपयोगी हुन्छन्। टपिक ब्रान्च भनेको छोटो समयका लागि बनाइने ब्रान्च हो, जुन तपाईँले कुनै एउटा विशेष फिचर वा सम्बन्धित कामका लागि प्रयोग गर्नुहुन्छ।

Git मा, ब्रान्चहरू बनाउने, तिनमा काम गर्ने, मर्ज गर्ने, र डिलिट गर्ने प्रक्रिया सामान्यत: धेरै पटक एकै दिनमा गरिन्छ।  
यो प्रविधिले छिटो र पूर्ण रूपमा काम switch गर्न सजिलो बनाउँछ।

जस्तै, तपाईँ master शाखामा काम गर्दै हुनुहुन्छ र एउटा समस्या समाधान गर्न नयाँ शाखा (जस्तै iss91) बनाउनुहुन्छ। त्यसमा केही समय काम गरेर, अर्को तरिका परीक्षण गर्न **iss91v2** शाखा बनाउनुहुन्छ। त्यसपछि, **master** शाखामा फर्केर केही काम गर्नुहुन्छ र नयाँ ब्रान्च (dumbidea) मा एउटा प्रयोगात्मक विचार परीक्षण गर्नुहुन्छ।

यसरी, तपाईँको commit इतिहास बिभिन्न ब्रान्चहरूमा भिन्न-भिन्न प्रकारका कामहरू देखाउने गरी संरचित हुन्छ।

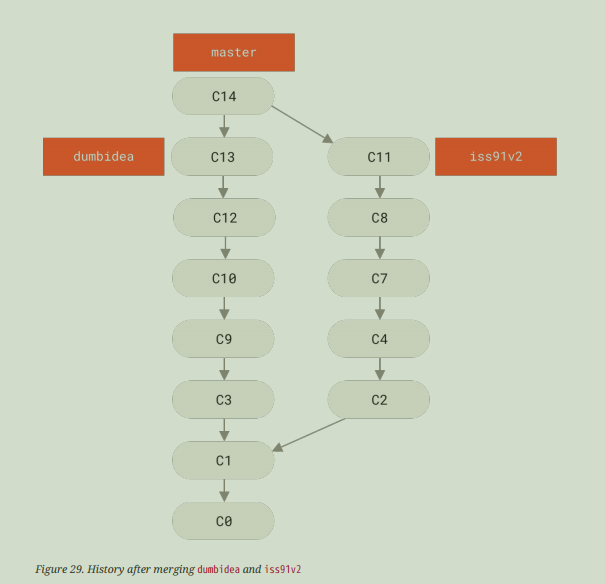


अब, मानौं तपाईंलाई दोस्रो समाधान (iss91v2) सबैभन्दा राम्रो लाग्यो, र तपाईंले dumbidea शाखा आफ्ना सहकर्मीहरूलाई देखाउनुभयो, जसले यसलाई एकदमै राम्रो विचार माने। तपाईं सुरुमा बनाएको iss91 शाखा हटाउन सक्नुहुन्छ (C5 र C6 commits हराउने गरी) र बाँकी दुई शाखा merge गर्न सक्नुहुन्छ।

यसपछि, तपाईंको commit इतिहास यस्तो देखिन्छ:

* तपाईंसँग अब केवल ती कामहरू हुनेछन्, जुन तपाईंले महत्वपूर्ण ठानेर merge गर्नुभयो।
* पुरानो शाखाको कुनै पनि अनावश्यक काम हराउँछ।

यसले तपाईको इतिहासलाई सफा, स्पष्ट, र व्यवस्थापन गर्न सजिलो बनाउँछ।



हामी Distributed Git मा Git प्रोजेक्टका विभिन्न सम्भावित workflow हरूको विस्तृत जानकारी दिनेछौं। त्यसैले, आफ्नो अर्को प्रोजेक्टमा कुन branching scheme प्रयोग गर्ने निर्णय गर्नु अघि, त्यो अध्याय अवश्य पढ्नुहोस्।

यो कुरा याद गर्न महत्त्वपूर्ण छ कि यी शाखाहरू पूर्ण रूपमा **स्थानीय** हुन्छन्।  
जब तपाईं branching र merging गर्दै हुनुहुन्छ, सबै काम केवल तपाईंको Git repository भित्र मात्र भइरहेको हुन्छ — server संग कुनै पनि संवाद हुँदैन।