Verzovací systémy a jejich použití

Verzování

Znáte nástroje pro verzování? Používali jste nějaký?

Co je to Git?

- **distribuovaný systém správy verzí** určený pro sledování změn v souborech
- koordinaci práce mezi lidmi na různých verzích těchto souborů
- Nástroj, který umožňuje vám a vašemu týmu pracovat na projektech a udržovat přehled o tom, co bylo změněno, kdo to udělal a kdy

Základní princip

- zaznamenává historii změn ve formě **commitů**, které obsahují snímek stavu vašeho projektu v daném okamžiku
- Každý vývojář má svou vlastní kopii celého repozitáře. To umožňuje **nezávislou** práci a následně **sloučení změn** z různých kopií repozitáře.

Proč používat Git ?

- Sledování změn
- Spolupráce
- Větvení
- Zpětné vrácení změn
- Ochrana dat
- Podpora pro týmy
- Open source a popularita (proč si myslíte, že je to dobré?)
- Flexibilita
- Rychlost

Git ve známých projektech

- Linux kernel
- Git
- Ruby on rails
- Trinity Core (WoW server)
- ASP.NET Core MVC
- Riot Games - tvůrci LoL

Pro zajímavost - nemusíte si pamatovat

Trocha historie

Linus Torvalds, 2005, ještě student, BitKeeper

První oficiální verze Gitu byla uvolněna 7. dubna 2005. Otcem Gitu je Linus Torvalds, jeden z hlavních vývojářů Linux kernelu. Původně byl Linux kernel vyvíjen za pomoci Bitkeeperu, ale společnost tento produkt zpoplatnila, a tak byli vývojáři Linuxu přinuceni vyvinout vlastní nástroj pro správu verzí. Tohoto úkolu se ujal již zmiňovaný Linus Torvalds.

Využití v praxi

- jednoduše zákazníkům publikovat náš projekt ve **verzích**, které nám vyhovují
- Díky **větvení** můžeme vytvářet novou funkcionalitu, aniž by to mělo jakýkoliv dopad na zbytek projektu.

Git však není omezen pouze na správu zdrojového kódu. Jeho schopnosti ve verzování jsou užitečné také při psaní odborných prací, jako jsou **ročníkové** nebo **maturitní** práce. Můžeme ho tedy pohodlně využít i v oblastech mimo vývoj softwaru.

Distribuovaný systém správy verzí

- Zdrojový kód je roz**distribuován** mezi jednotlivé vývojáře. Každý vývojář má **vlastní kopii** projektu. (distribuovaný)
- Nezávislý na dalších nástrojích. Git lze zkompilovat pro jakoukoli platformu. Není závislý na žádných externích knihovnách (systém)
- Git pomáhá se **správou** projektu nezávisle na jeho velikosti. (správa)
- Stav projektu ukládáme do **verzí**. Jednoduchý přístup k celé **historii projektu.** (verzí)

Opakem distribuovaného je **centralizovaný**, kterým je například SVN. Konkurenční SVN má **jeden serve**r a pokud chce vývojář provádět změny, musí být k tomuto serveru připojen

Verzovací systémy a jejich použití

Reálný princip

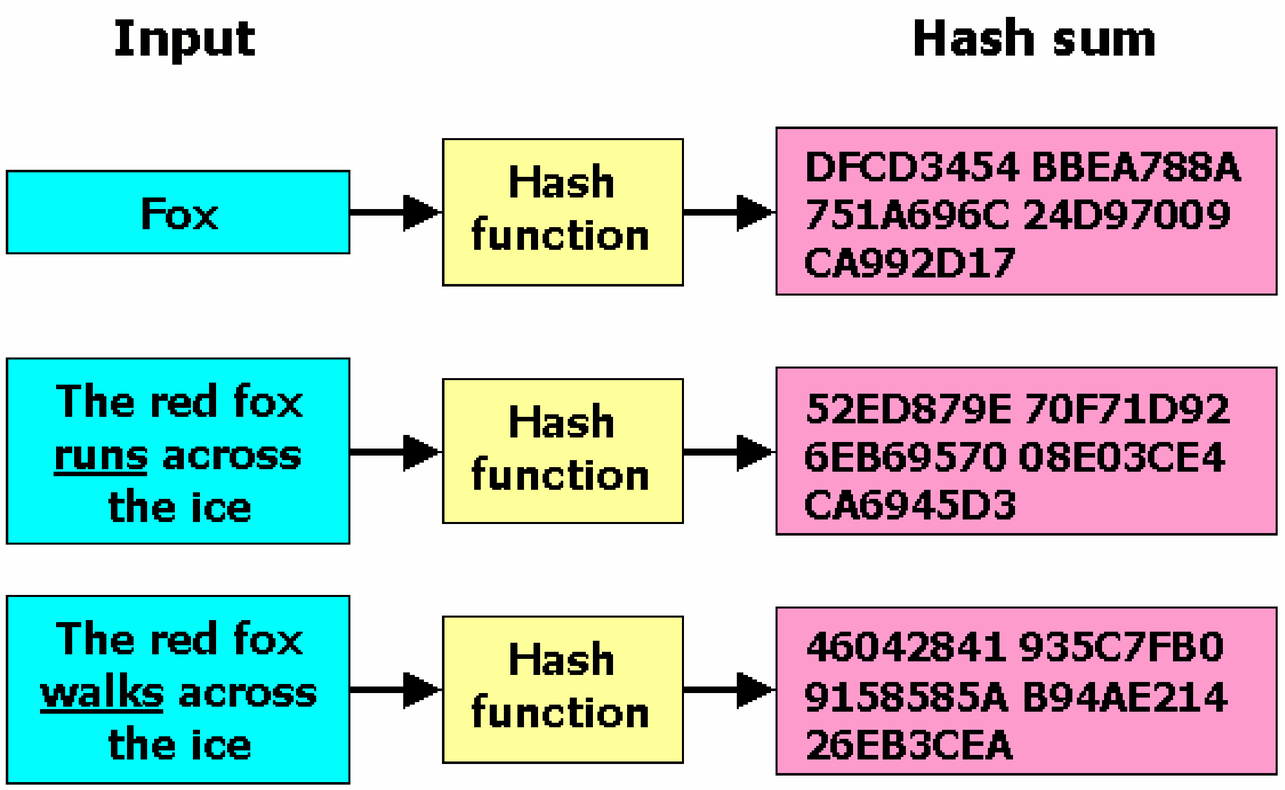
Co je základní jednotkou? Co je to commit? Jaké příkazy si pamatujete?

Princip

- **Repositář** místo, které slouží k ukládání a udržování celé historie projektu.
- Každý commit (uložený stav projektu v určitém okamžiku vývoje) je zaznamenán v repositáři spolu s informacemi o tom, **kdo provedl změnu, kdy a co konkrétně bylo změněno**
- Git každý soubor uloží pouze jednou a poté ukládá jen tzv. **snapshoty**.
- Ukládá všechny soubory **binárně.** *Proč?*
- Každá operace, kterou uděláme, je **nejdříve lokální.**
- => Není prakticky nic, co by vám Git nedovolil bez připojení k internetu.

Princip

- **Integrita**. Pro každý soubor je kontrolní součet. SHA-1 hash (40-ti místné hexadecimální číslo)
- Nemůžeme tedy změnit nebo poškodit soubor, aniž by o tom Git nevěděl. Git interně neukládá soubory podle jména, ale právě podle tohoto kontrolního součtu.
- Git pouze přidává data. I když vymažeme řádek, Git pouze dostane informaci o přidání „ničeho“ do souboru
- Rozdělen do čtyř prostorů



Jako **hash** označují vývojáři krátký řetězec písmen a číslic, který vznikne tím, že proženeme jiný řetězec (vstupní data) tzv. **hashovací funkcí.**

K čemu je to dobrý? Viděli jste to někde? Jaké znáte?

```
$ git log
commit 7893dd38bd467d8397205da661a6f7a59ef77032
Author: sagar <sagar@softpost.com>
Date:   Sun Jun 19 08:12:54 2016 +1000

    committing f1.txt

commit 5631fb6061d01ddeec8e4226a39dabb3f2fdd122
Author: sagar <sagar@softpost.com>
Date:   Sun Jun 12 14:59:05 2016 +1000

    repo version

commit 574738895988l4ec150baeb36a1d7997946b27a4
Author: sagar <sagar@softpost.com>
Date:   Sun Jun 12 14:20:11 2016 +1000

    Commiting gitignore

commit 5d988010196cf5fb2cc60ae6cf31b6dff3b60f3a
Author: sagar <sagar@softpost.com>
Date:   Sun Jun 12 13:58:11 2016 +1000

    First commit
```

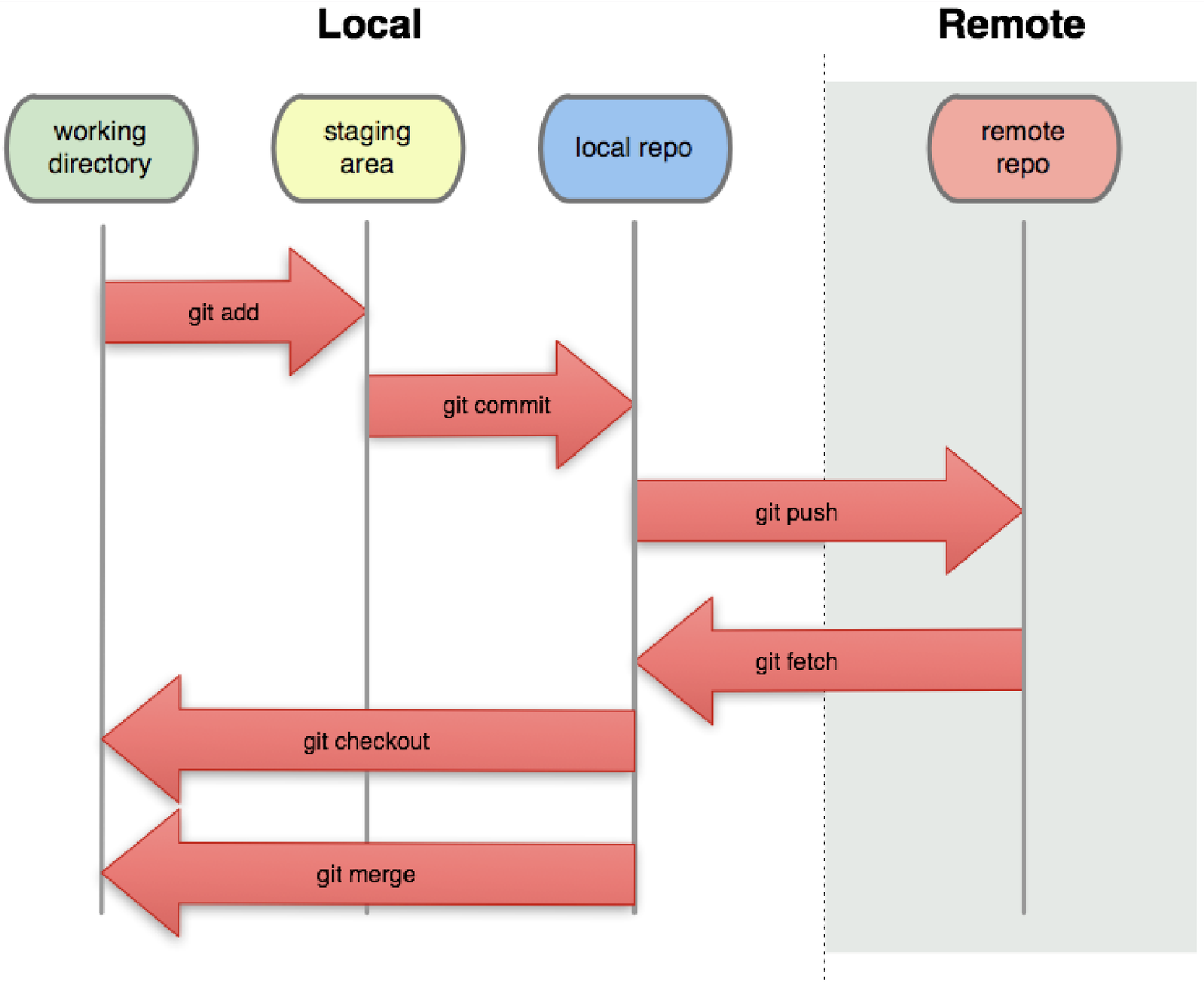
Staging area

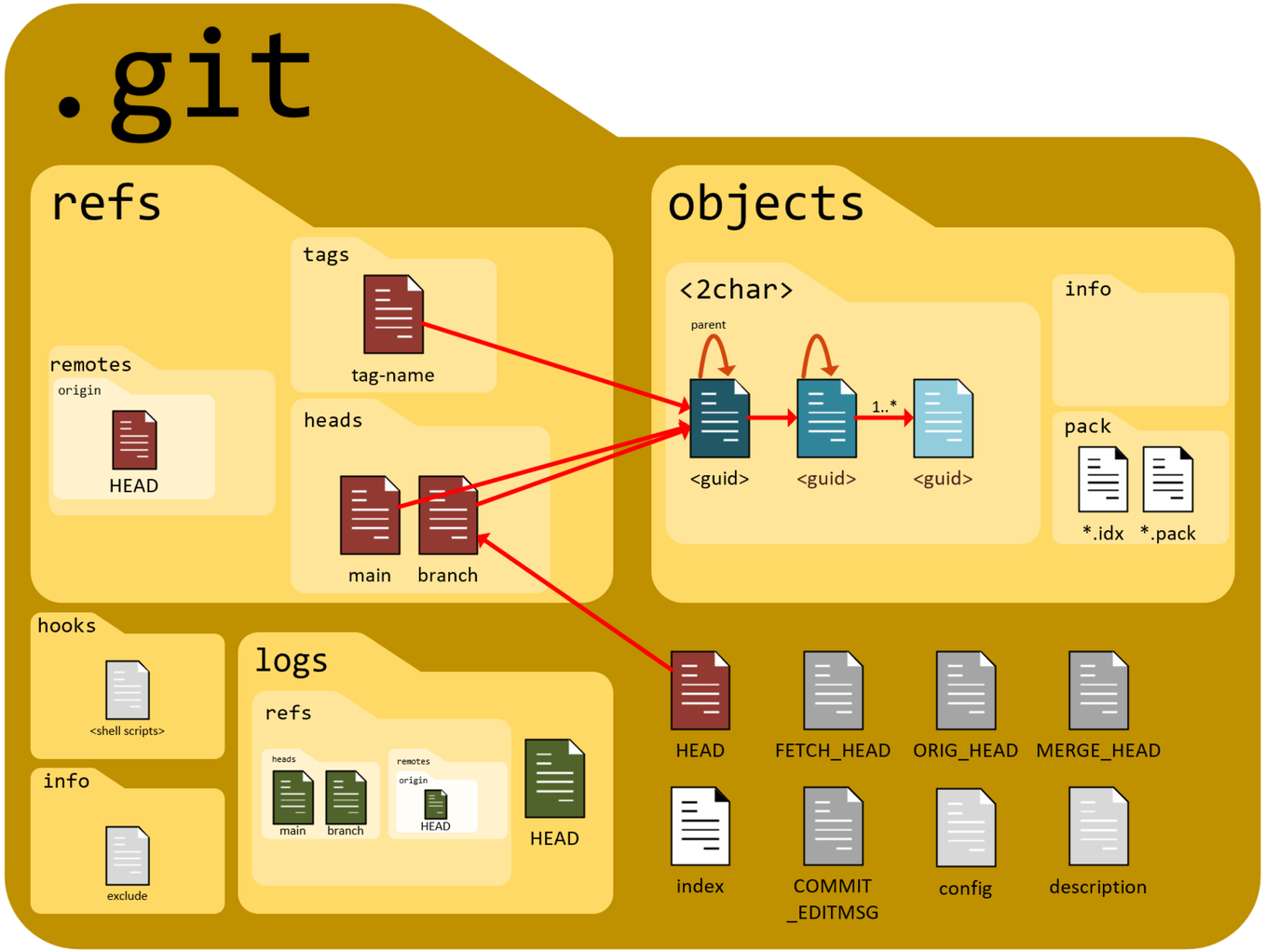
Vlastně se rozhodujete, jestli upravené soubory chcete zahrnout a nadepsat nějakým labelem.

To understand what is staging area is, let’s take a real-world example – suppose that you are moving to another place, and you have to pack your stuff into boxes and you wouldn’t want to mix the items meant for the bathroom, kitchen, bedroom, and the living room in the same box. So, you will take a box and start putting stuff into it, and if doesn’t make sense, you can also remove it before finally packing the box and labeling it.

Here, in this example, the box serves as the staging area, where you are doing the work (crafting your commit), whereas when you are done, then you are packing it and labeling it (committing the code).

https://www.developernation.net/blog/git-internals-part-3-understanding-the-staging-area-in-git


Local
Remote
working directory
staging area
local repo
remote repo
git add
git commit
git push
git fetch
git checkout
git merge


.git
refs
tags
tag-name
remotes
origin
HEAD
heads
main
branch
objects
<2char>
parent
<guid>
<guid>
1..*
<guid>
info
pack
*.idx
*.pack
hooks
<shell scripts>
info
exclude
logs
refs
heads
main
branch
remotes
origin
HEAD
HEAD
HEAD
FETCH_HEAD
ORIG_HEAD
MERGE_HEAD
index
COMMIT
_EDITMSG
config
description
commit
tree
blob
object
configuration
temp
log
ref

```
├── COMMIT_EDITMSG                  # A text file, the annotation of the last commit
├── FETCH_HEAD                      # A text file, the SHA-1 hash of the last commit that communicated with the server in each partition
├── HEAD                            # A text file, the branch where the current workspace locates
├── ORIG_HEAD                       # A text file, synchronizing the SHA-1 hash of the last commit in the current branch with that in the remote branch.
├── branches
├── config                          # Git-related configuration, such as remote addresses, branch pointers, policies, and modes
├── description
├── hooks                           # Hooks, related hooks that are exposed by Git in different lifecycle stages
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── prepare-commit-msg.sample
│   └── update.sample
├── index                           # Index, index files that can be viewed by using the git ls-files command
├── info
│   ├── exclude
│   └── refs
├── logs                            # Commit log, existing in each branch; the reason why Git can log offline
│   ├── HEAD
│   └── refs
│       ├── heads
│       └── remotes
├── objects                         # One of the most important file storage directories
│   ├── b4                          # Folders that store loose files; folder names being the first two characters of SHA-1 hashes
│   │   └── 6fc1c643a89564f88decc6be9cf374153017bc
│   ├── info                        # Directory information, such as the name of the pack file
│   │   └── packs
│   └── pack                        # Related *.idx and *.pack files are generated after loose files are packed.
│       ├── pack-21351cd8ce7bca1fa90299498acffb3890232dda.idx
│       └── pack-21351cd8ce7bca1fa90299498acffb3890232dda.pack
├── packed-refs                     # The files in the refs directory are packed into a file, which is usually the last modification set of each branch.
├── refs                            # The last modification set of each branch, which is a snapshot.
│   ├── heads
│   │   └── daily
│   ├── remotes
│   │   └── origin
│   └── tags
```

Čtyři pracovní prostory

Pracovní složka (Working Directory). aktuální verzi vašeho projekt. Provádíme zde úpravy, vytváříme, mažeme a upravujeme soubory **podle svých potřeb**.

Staging area. Po provedení určitých změn, vybíráme soubory do dalšího commitu. Tento prostor slouží jako index, které změny budou součásti dalšího commitu.

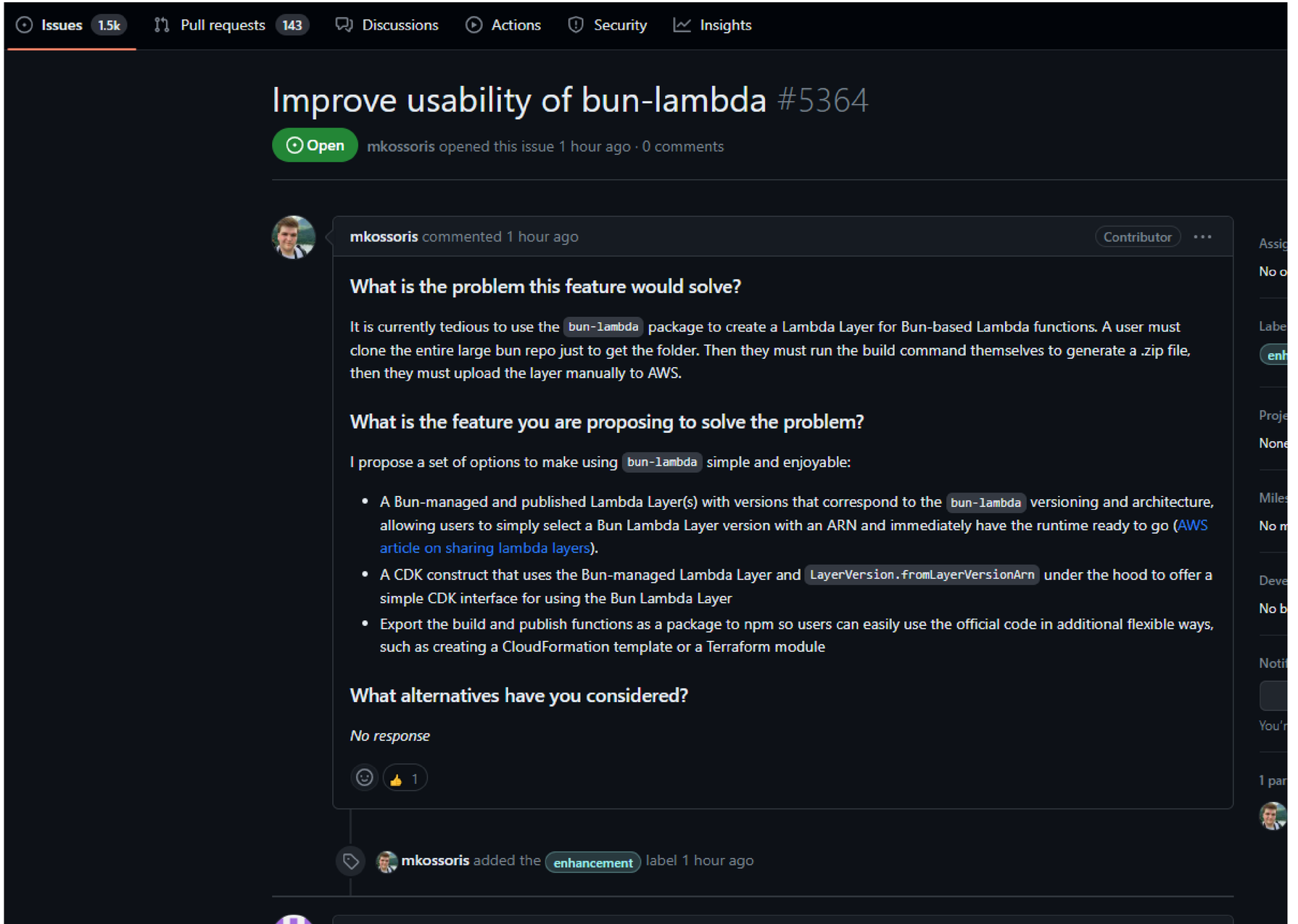
$pvacha: git add .

Local repo. Soubory ze staging area - commit, což je trvalé uložení těchto změn do lokálního repozitáře. Je zde kompletní historie našeho projektu včetně všech změn a commitů, které jsme provedli.

$pvacha: git commit -m “Změna”

Remote repo. Repozitář umístěný na vzdáleném serveru. Změny viditelné ostatním uživatelům. Tím se umožňuje sdílení a koordinace práce mezi různými vývojáři.

$pvacha: git push



Issues 1.5k Pull requests 143 Discussions Actions Security Insights

Improve usability of bun-lambda #5364

Open mkossoris opened this issue 1 hour ago · 0 comments

mkossoris commented 1 hour ago Contributor

What is the problem this feature would solve?

It is currently tedious to use the `bun-lambda` package to create a Lambda Layer for Bun-based Lambda functions. A user must clone the entire large bun repo just to get the folder. Then they must run the build command themselves to generate a .zip file, then they must upload the layer manually to AWS.

What is the feature you are proposing to solve the problem?

I propose a set of options to make using `bun-lambda` simple and enjoyable:

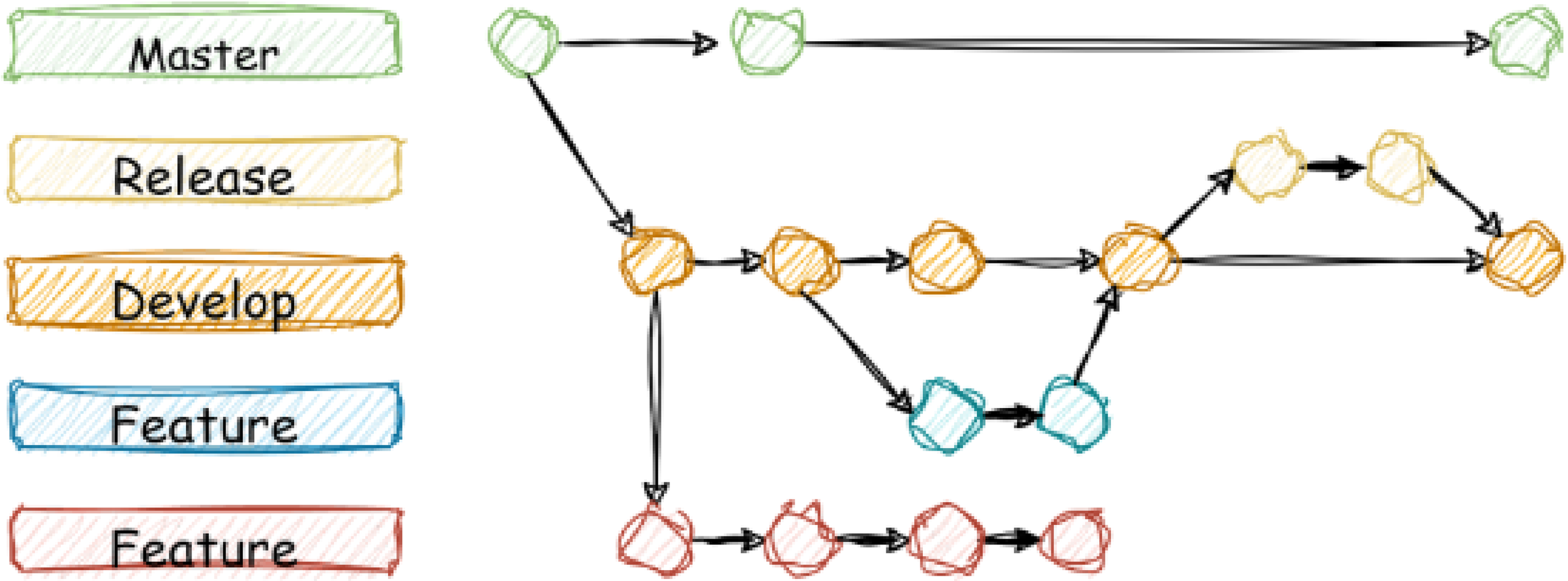
- A Bun-managed and published Lambda Layer(s) with versions that correspond to the `bun-lambda` versioning and architecture, allowing users to simply select a Bun Lambda Layer version with an ARN and immediately have the runtime ready to go (AWS article on sharing lambda layers).
- A CDK construct that uses the Bun-managed Lambda Layer and `LayerVersion.fromLayerVersionArn` under the hood to offer a simple CDK interface for using the Bun Lambda Layer
- Export the build and publish functions as a package to npm so users can easily use the official code in additional flexible ways, such as creating a CloudFormation template or a Terraform module

What alternatives have you considered?

No response

1

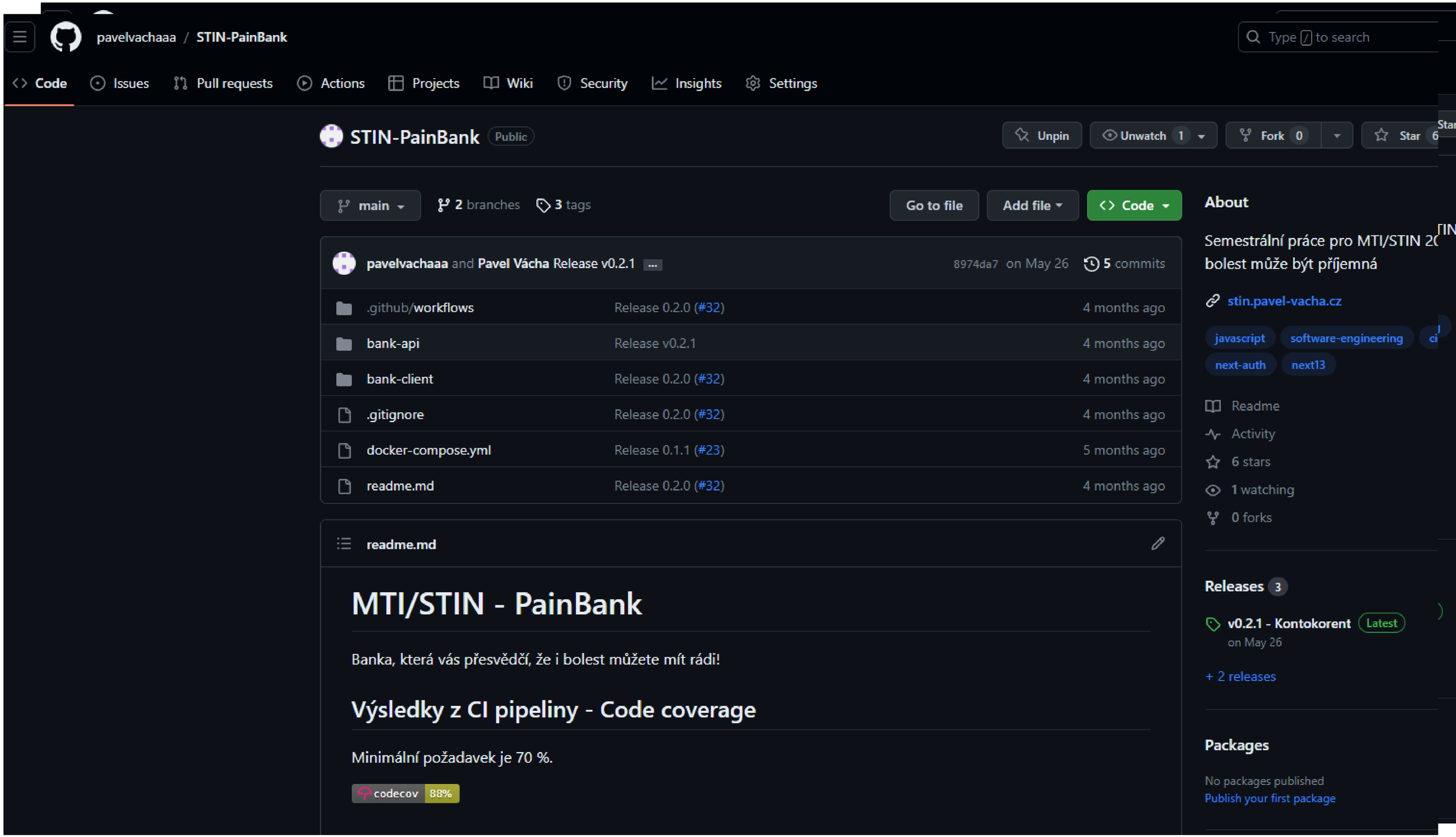
mkossoris added the enhancement label 1 hour ago


Master
Release
Develop
Feature
Feature

Jak nemá vypadat commit?

Poskytovatelé vzdálených repozitářů

- **GitHub**
- GitLab
- Bitbucket
- Azure DevOps
- SourceForge
- Beanstalk
- **Gitea** (kritická infrastruktura)


pavelvachaaa / STIN-PainBank
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
STIN-PainBank
Public
Unpin
Unwatch 1
Fork 0
Star 6
main
2 branches
3 tags
Go to file
Add file
Code
pavelvachaaa and Pavel Vácha Release v0.2.1
8974da7 on May 26
5 commits
.github/workflows
Release 0.2.0 (#32)
4 months ago
bank-api
Release v0.2.1
4 months ago
bank-client
Release 0.2.0 (#32)
4 months ago
.gitignore
Release 0.2.0 (#32)
4 months ago
docker-compose.yml
Release 0.1.1 (#23)
5 months ago
readme.md
Release 0.2.0 (#32)
4 months ago
readme.md
MTI/STIN - PainBank
Banka, která vás přesvědčí, že i bolest můžete mít rádi!
Výsledky z CI pipeliny - Code coverage
Minimální požadavek je 70 %.
codecov 88%
About
Semestrální práce pro MTI/STIN 2
bolest může být příjemná
stin.pavel-vacha.cz
javascript
software-engineering
next-auth
next13
Readme
Activity
6 stars
1 watching
0 forks
Releases 3
v0.2.1 - Kontokorent Latest
on May 26
+ 2 releases
Packages
No packages published
Publish your first package

Project

GitLab
Manage
Plan
Code
Build
Deploy
Operate
Monitor
Analyze

GitLab

Project ID: 278964

Star 4627

358,811 Commits | **15,821** Branches | **2,217** Tags | **67.5 TiB** Project Storage | **139** Releases | **4,344** Environments

Topics: hacktoberfest | Ruby | Vue.js | + 1 more

GitLab is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. Self-host GitLab on your own servers, in a container, or on a cloud provider.

pipeline running | Ruby Coverage unknown | JS Coverage 66.71% | Contribute community fork

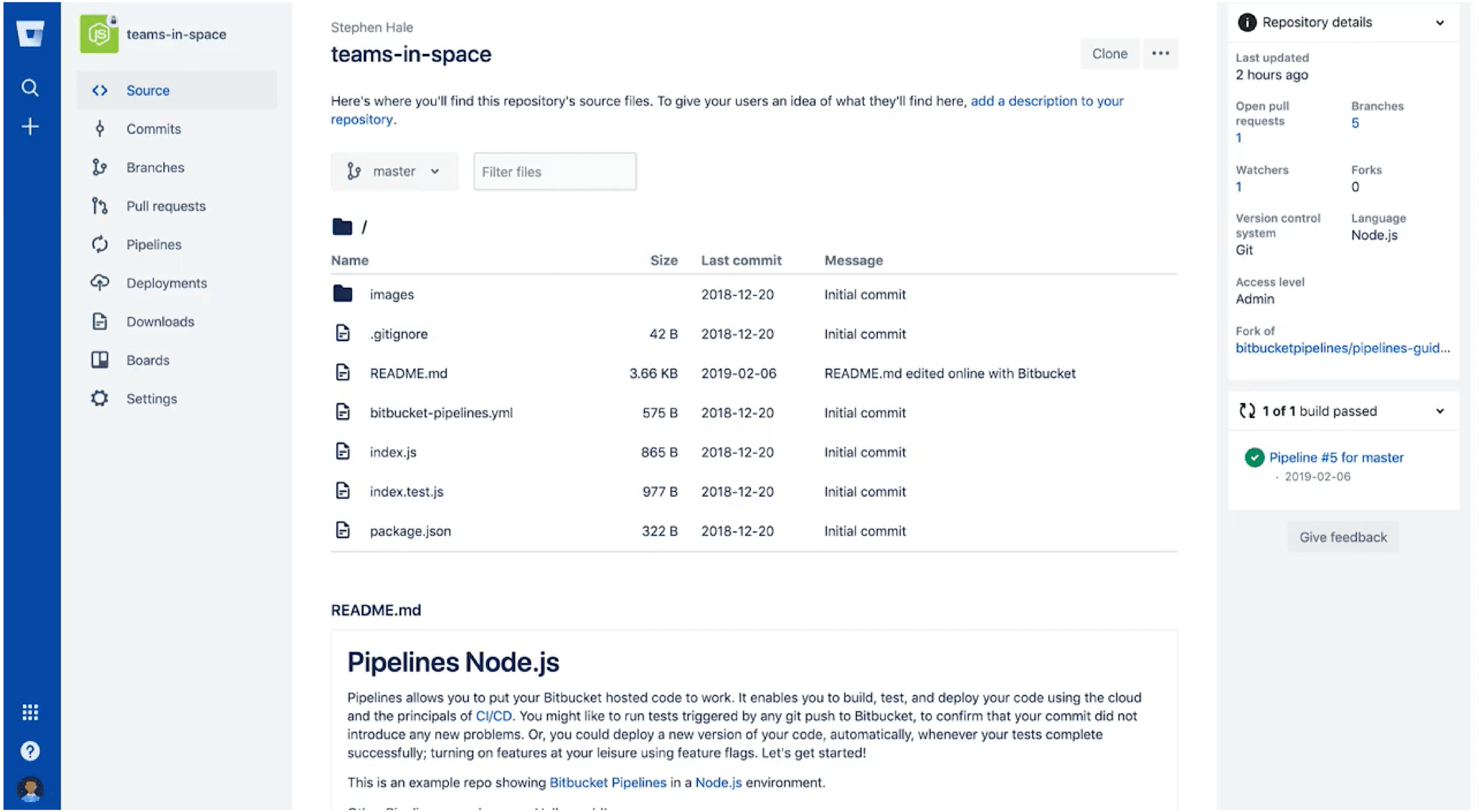
Merge branch 'pl-unlimited_max_formatted_output_length' into 'master'
Mario Celi authored 30 minutes ago

2cd2cd87

master | gitlab | History | Find file | Clone

README | MIT License | CHANGELOG | CONTRIBUTING

Name	Last commit	Last update
.github	Rename GitLab CE to FOSS in GitHub issue templates	3 years ago
.gitlab	Merge branch 'cablett-danger-fix' into 'master'	2 hours ago
.lefthook/pre-push	Add security harness to Lefthook	5 months ago


teams-in-space
Source
Commits
Branches
Pull requests
Pipelines
Deployments
Downloads
Boards
Settings
Stephen Hale
teams-in-space
Clone
Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.
master
Filter files
/
Name
Size
Last commit
Message
images
2018-12-20
Initial commit
.gitignore
42 B
2018-12-20
Initial commit
README.md
3.66 KB
2019-02-06
README.md edited online with Bitbucket
bitbucket-pipelines.yml
575 B
2018-12-20
Initial commit
index.js
865 B
2018-12-20
Initial commit
index.test.js
977 B
2018-12-20
Initial commit
package.json
322 B
2018-12-20
Initial commit
README.md
Pipelines Node.js
Pipelines allows you to put your Bitbucket hosted code to work. It enables you to build, test, and deploy your code using the cloud and the principals of CI/CD. You might like to run tests triggered by any git push to Bitbucket, to confirm that your commit did not introduce any new problems. Or, you could deploy a new version of your code, automatically, whenever your tests complete successfully; turning on features at your leisure using feature flags. Let's get started!
This is an example repo showing Bitbucket Pipelines in a Node.js environment.
Repository details
Last updated
2 hours ago
Open pull requests
1
Branches
5
Watchers
1
Forks
0
Version control system
Git
Language
Node.js
Access level
Admin
Fork of
bitbucketpipelines/pipelines-guid...
1 of 1 build passed
Pipeline #5 for master
2019-02-06
Give feedback

Příští teoretická a praktická hodina

Co nás čeká a nemine?

Čekají nás celkem ještě **tři** plné vyučovací hodiny s verzovacími systémy.

Poté začneme úvod do **programovacích jazyků a databází.**

Jo... A **test** z verzovacích systému, jakmile je dokončíme. :((