Universiteti i Prishtinës "Hasan Prishtina" Kosovë

Aplikacione kompjuterike I

Pjesa 1 - Hyrje

Prof. Asoc. Dr. Ermir Rogova

Objektivat

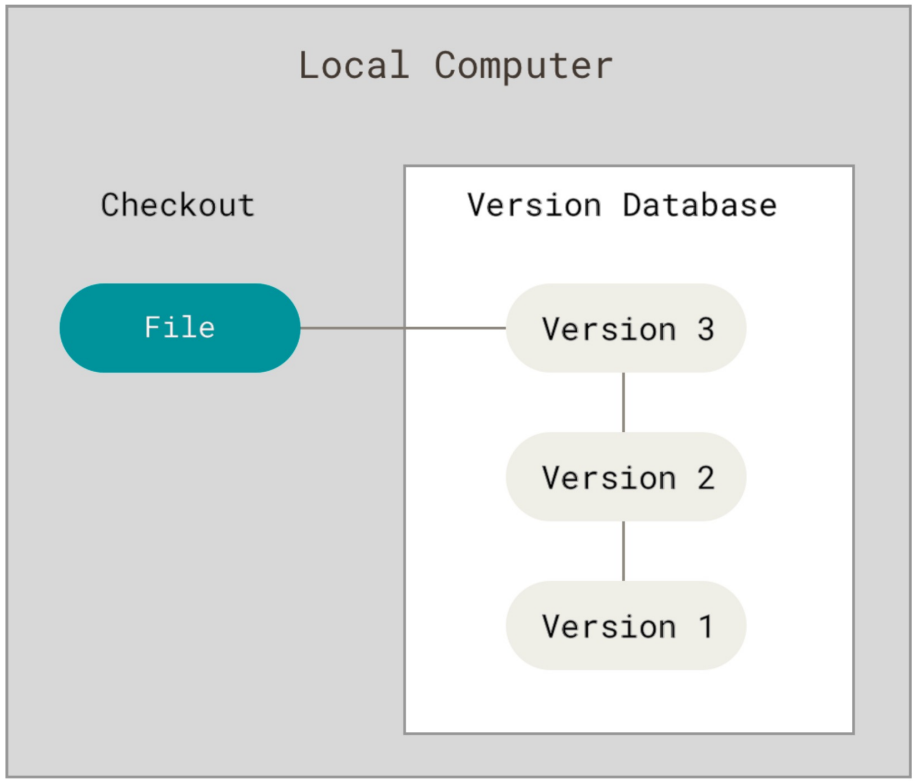
- Ky kapitull ka të bëjë me fillimin e punës me Git.
- Njoftim me mjetet e kontrollit të versionit (version control tools),
- Git në sistemin tuaj,
- Konfigurimi për të filluar punën me Git
- Me përfundimin e këtij kapitulli ju do të kuptoni pse Git egziston, pse duhet ta përdorni dhe duhet të jeni gati për ta përdorur.

Çfarë është kontrolli i versionit

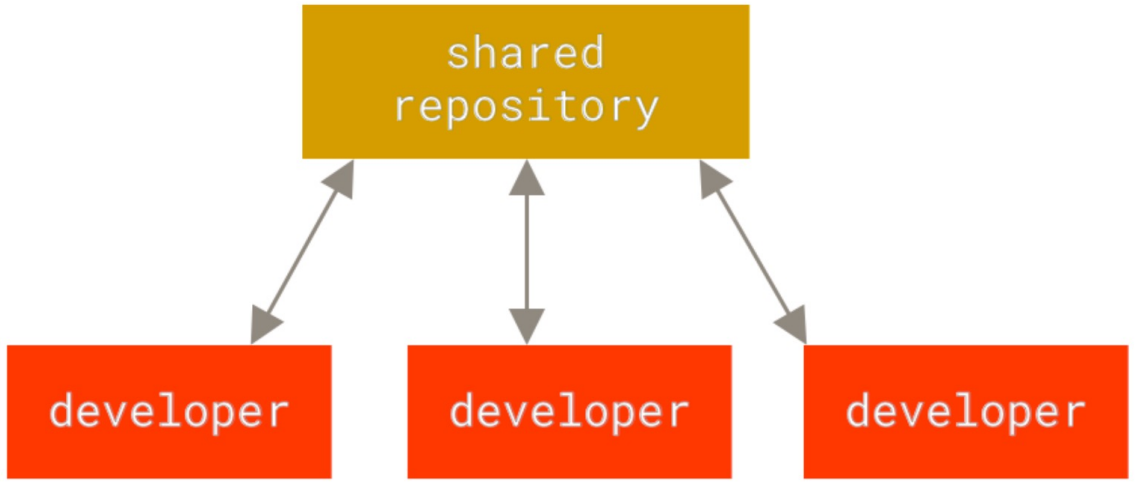
- Kontrolli i versionit është një sistem që regjistron ndryshimet në një fajl ose grup fajlash me kalimin e kohës, në mënyrë që të mund të ri-ktheni versione specifike më vonë.
- Nëse p.sh. jeni një dizajner grafik ose web dizajner dhe dëshironi të mbani çdo version të një imazhi ose shtritje (layout), një Sistem i Kontrollit të Versionit (VCS) është një gjë shumë e mençur për t'u përdorur.
- Ky ju mundëson:
  - të ktheni fajlat e përzgjedhur në një gjendje të mëparshme,
  - të ktheni të gjithë projektin në një gjendje të mëparshme,
  - të krahasoni ndryshimet me kalimin e kohës,
  - të shihni se kush ka modifikuar për herë të fundit diçka që mund të shkaktojë një problem,
  - kush ka paraqitur një problem dhe kur,
  - dhe më shumë.
- Përdorimi i një VCS gjithashtu do të thotë që nëse ndodhin dëmtime ose humbni fajlat, mund ti riktheni lehtësisht.
- Të gjitha këto përfitime merren për shumë pak mund.

VCS lokale

- Metoda e preferuar e kontrollit të versionit për shumë njerëz është kopjimi i fajlave në një folder tjetër (ndoshta një folder me vulë kohore).
- Kjo qasje është shumë e zakonshme sepse është shumë e thjeshtë, por është gjithashtu tepër e prirur për gabime.
- Është e lehtë të harrosh se në cilën folder ndodhesh dhe të shkruash aksidentalisht në skedarin e gabuar ose të kopjosh mbi skedarë që nuk dëshiron.
- Për t'u marrë me këtë çështje, programuesit shumë kohë më parë zhvilluan VCS lokale që kishin një bazë të dhënash të thjeshtë që mbante të gjitha ndryshimet në fajla.
- Një nga mjetet më të njohura të VCS ishte një sistem i quajtur RCS, i cili ende përdoret në shumë kompjuterë sot.
- RCS (Revision Control System) funksionon duke mbajtur grupet e patch-eve (d.m.th., dallimet midis fajlave) në një format të veçantë në disk; më pas mund të rikrijojë se si dukej çdo fajl në çdo moment në kohë duke i bërë bashkë të gjitha pjesët.



VCS e centralizuara



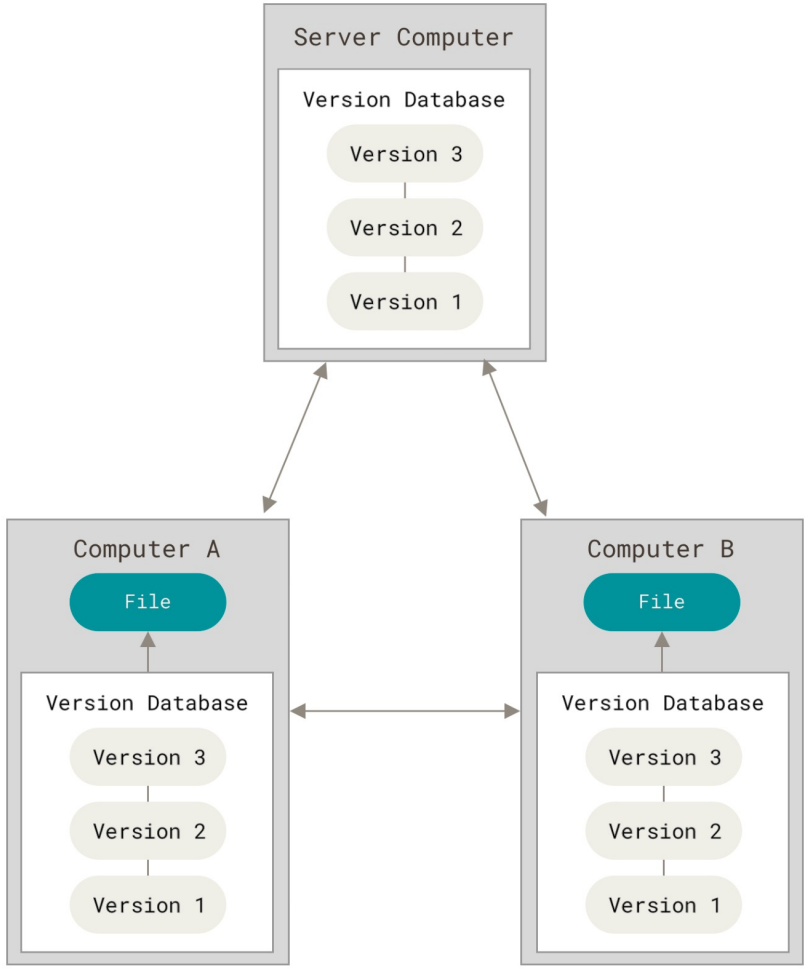
- Çështja tjetër kryesore që hasim është se shpesh duhet të bashkëpunojnë me persona të tjerë.
- Për t'u marrë me këtë problem, u zhvilluan Sistemet e Kontrollit të Versionit të centralizuara (CVCS).
- Këto sisteme (të tilla si CVS, Subversion dhe Perforce) kanë një server të vetëm që përmban të gjithë fajlat e versionuar dhe një numër klientësh që punojnë me fajlat nga ai lokacion qendror.
- Për shumë vite, ky ka qenë standardi për kontrollin e versioneve.

VCS e centralizuara

- Ky konfigurim ofron shumë përparësi, veçanërisht mbi VCS-të lokale.
  - Të gjithë e dinë në një masë të caktuar se çfarë po bëjnë të gjithë të tjerët në projekt.
  - Administratorët kanë kontroll të imët se kush mund të bëjë çfarë, dhe është shumë më e lehtë të administrosh një CVCS sesa të merresh me bazat e të dhënave lokale për çdo klient.
- Megjithatë, ky konfigurim ka edhe disa dobësi serioze.
  - Më e dukshme është pika e vetme e dështimit që përfaqëson serveri i centralizuar.
  - Nëse ai server nuk funksionon për një orë, atëherë gjatë asaj ore askush nuk mund të bashkëpunojë fare ose të ruajë ndryshimet e versionuara në çdo gjë për të cilën po punon.
  - Nëse hard disku në të cilin ndodhet baza e të dhënave qendrore prishet dhe nuk janë mbajtur kopjet e duhura, ju humbni absolutisht gjithçka - të gjithë historinë e projektit, përveç çfarëdo fotografie të vetme që njerëzit kanë në makinat e tyre lokale.
  - VCS-të lokale vuajnë nga i njëjti problem - sa herë që keni të gjithë historinë e projektit në një vend të vetëm, rrezikoni të humbni gjithçka.

VCS e shpërndara (DVCS)

- Në një DVCS (si Git, Mercurial, Bazaar ose Darcs), klientët nuk kontrollojnë vetëm versionin më të fundit të fajlëve; përkundrazi, ata pasqyrojnë plotësisht depon, duke përfshirë historinë e tij të plotë.
- Kështu, nëse ndonjë server dëmtohet dhe këto sisteme po bashkëpunonin nëpërmjet atij serveri, çdo depo e klientit mund të kopjohet përsëri në server për ta rikthyer atë.
- Çdo klon është një kopje rezervë e plotë e të gjitha të dhënave.



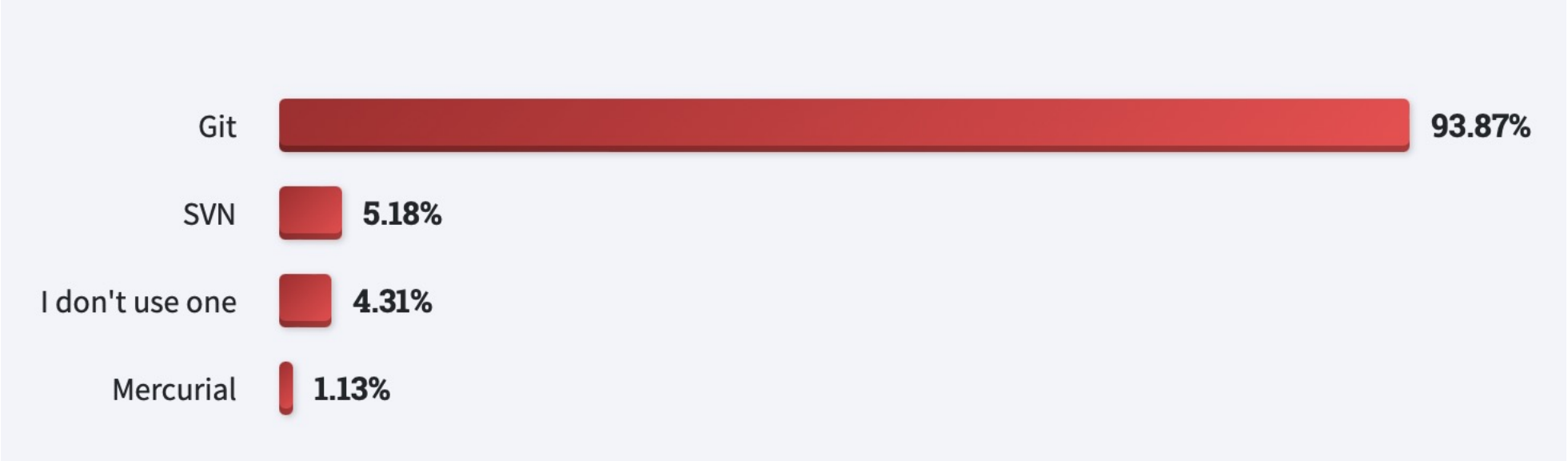
VCS e shpërndara (DVCS)

- Për më tepër, shumë nga këto sisteme funksionojnë mjaft mirë me më shumë se një depo të largët, kështu mundësohet bashkëpunimi me grupe të ndryshme njerëzish në mënyra të ndryshme njëkohësisht brenda të njëjtit projekt.
- Kjo lejon të vendosim disa lloje të rrjedhave të punës që nuk janë të mundshme në sistemet e centralizuara.

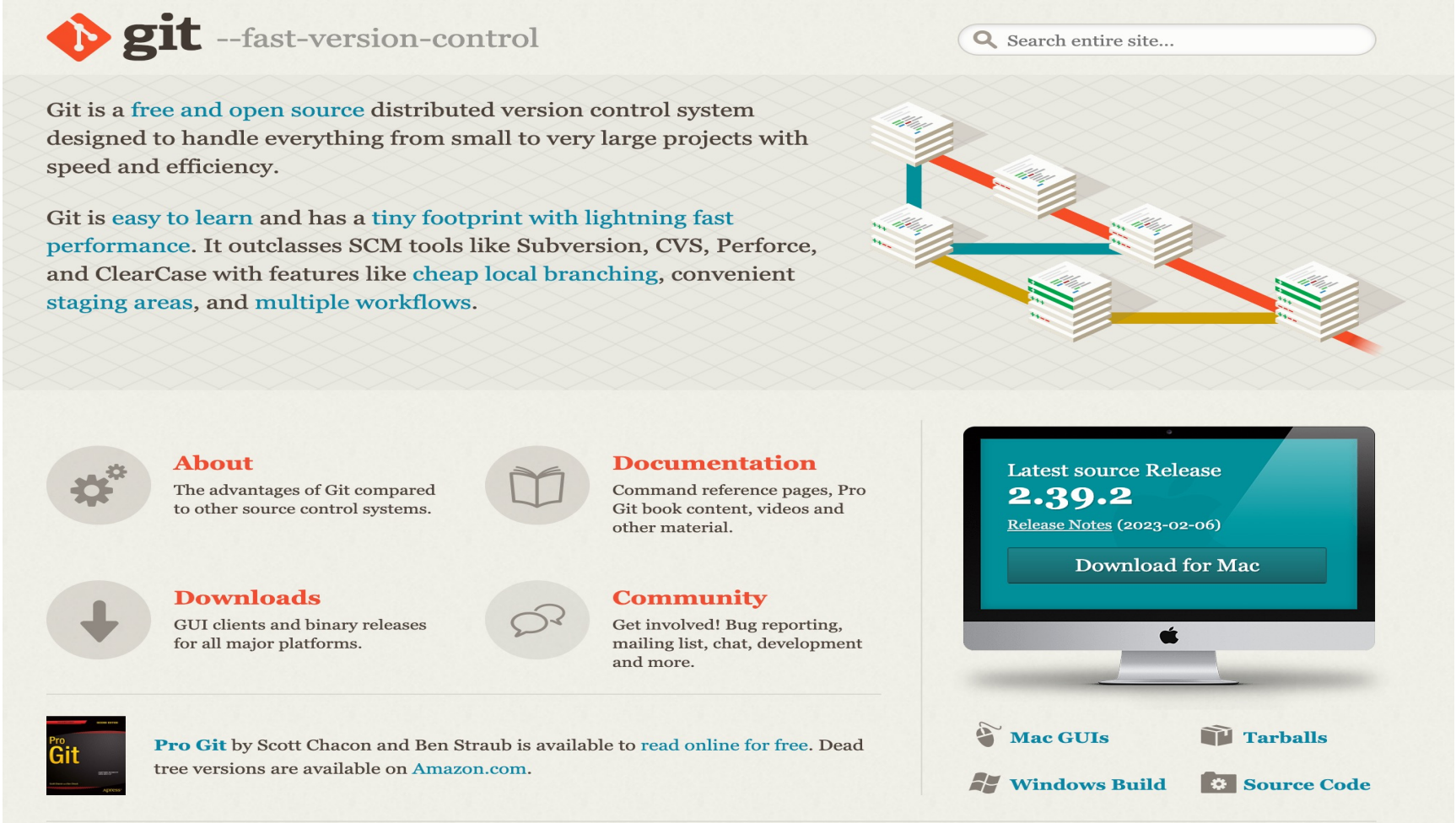
Pak histori për Git

- Linus Torvalds (krijuesi i Linux)
- Në 2005, ndërsa punonte në Linux u zhgënjye me sistemet e kontrollit të versioneve
- Ato ishin të ngadalta, me kod burimor të mbyllur dhe zakonisht me pagesë
- Vendosi të mos të vazhdojë përdorimin e BitKeeper dhe të krijojë vetë një VCS (Git).
- Më 3 prill 2005 ai filloi të punojë në Git.
- Disa nga qëllimet e sistemit të ri ishin si më poshtë:
  - Shpejtësia
  - Dizajn i thjeshtë
  - Mbështetje e fortë për zhvillimin jolinear (mijëra degë paralele)
  - Plotësisht i shpërndarë
  - Në gjendje të trajtojë me efikasitet projekte të mëdha si kerneli Linux (shpejtësia dhe madhësia e të dhënave)
- Brenda pak ditësh ai kishte kryer funksionet më themelore
- Lëshimi zyrtar i Git erdhi disa muaj më vonë
- Sot mbi 90% e zhvilluesve në mbarë botën përdorin Git në baza ditore

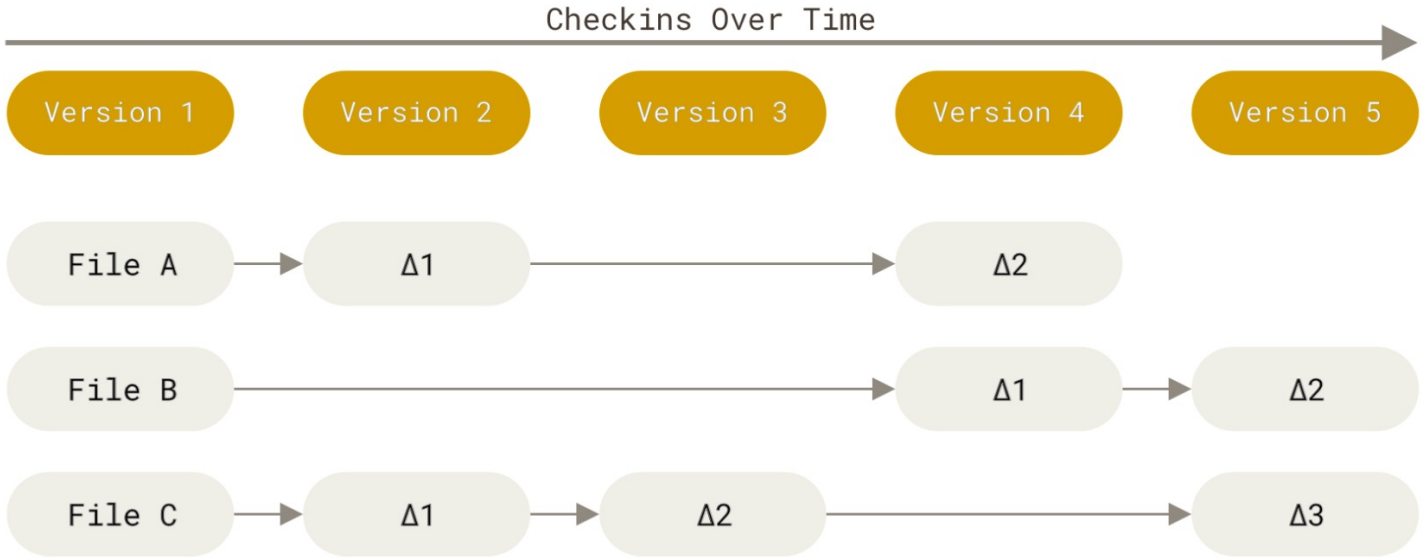
Anketa nga Stack overflow (2022)



Sipas tyre - https://git-scm.com/

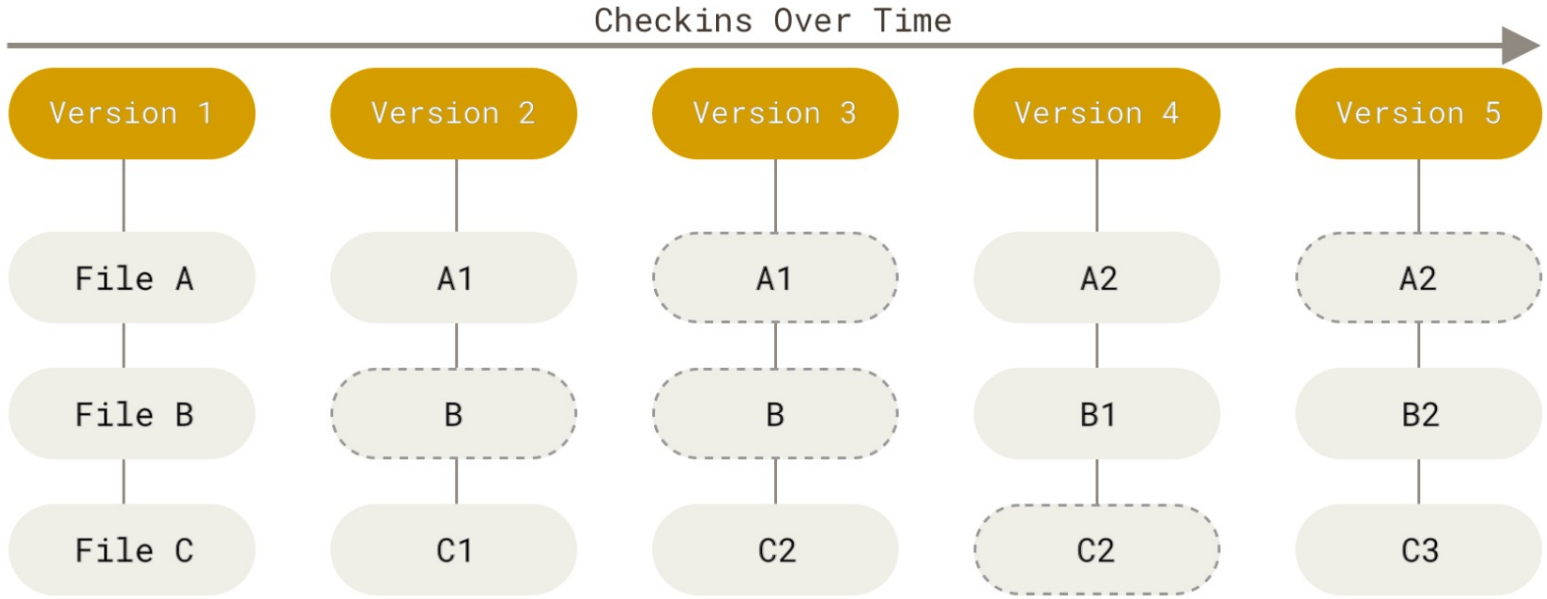


Git kundrejt të tjerëve



- Dallimi kryesor midis Git dhe çdo VCS tjetër është mënyra se si Git mendon për të dhënat e tij.
- Shumica e sistemeve të tjera ruajnë informacionin si një listë ndryshimesh të bazuara në fajla.
- Këto sisteme të tjera (CVS, Subversion, Perforce, Bazaar, etj) informacionin e ruajnë si një grup fajlësh bashkë me ndryshimet e bëra në secilin fajl me kalimin e kohës (kjo zakonisht përshkruhet si kontroll i versionit të bazuar në delta).

Git kundrejt të tjerëve



- Git shikon të dhënat e tij më shumë si një seri fotografish të një sistemi fajlash në miniaturë.
- Me Git, sa herë ruajmë gjendjen e projektit, Git në thelb merr një fotografi se si duken të gjithë fajlat në atë moment dhe ruan një referencë për atë fotografi.
- Për të qenë efikas, nëse fajlat nuk kanë ndryshuar, Git nuk e ruan përsëri fajlin, vetëm një link me fajlin e mëparshëm identik që ka ruajtur tashmë.
- Git mendon për të dhënat e tij më shumë si një **rrjedhë shkrepjesh** (stream of snapshots).

Operim lokal

- Shumica e operacioneve në Git kanë nevojë vetëm për fajla dhe burime lokale për të funksionuar - në përgjithësi nuk nevojitet asnjë informacion nga një kompjuter tjetër në rrjet.
  - Për shembull, për të shfletuar historinë e projektit, Git nuk ka nevojë të shkojë në server për të marrë historinë dhe për ta shfaqur atë - ai thjesht e lexon atë drejtpërdrejt nga baza e të dhënave lokale.
  - Nëse duam të shohim ndryshimet e paraqitura midis versionit aktual të një fajli dhe atij fajli një muaj më parë, Git mund të kërkojë fajlin e një muaji më parë dhe të bëjë një llogaritje lokale të diferencës.
- Kjo do të thotë poashtu se ka shumë pak gjëra që nuk mund të bëhen nëse jemi offline.

Integriteti

- Gjithçka në Git i nënshtrohet mbledhjes kontrolluese (checksum) përpara se të ruhet dhe më pas referohet nga ajo shumë kontrolli.
- Kjo do të thotë se është e pamundur të ndryshosh përmbajtjen e ndonjë fajli ose folderi pa u detektuar nga Git.
- Ky funksionalitet është i integruar në Git në nivelet më të ulëta dhe është pjesë përbërëse e filozofisë së tij.
- Nuk mund të humbasin informacione gjatë rrugës ose të korruptohen fajlat pa qenë në gjendje ta zbulojë Giti këtë.

SHA-1

- Mekanizmi që përdor Git për këtë mbledhje kontrolli quhet hash SHA-1.
- Ky është një varg 40 karakteresh i përbërë nga karaktere heksadecimal (0–9 dhe a–f) dhe i llogaritur bazuar në përmbajtjen e një fajli ose folderi në Git.
- Një hash SHA-1 duket si këtu: 24b9da6552252987aa493b52f8696cd6d3b00373
- Do të shohim këto vlera hash kudo në Git sepse i përdor ato aq shumë.
- Në fakt, Git ruan gjithçka në bazën e të dhënave të tij jo sipas emrit të fajlit, por sipas vlerës hash të përmbajtjes së tij.

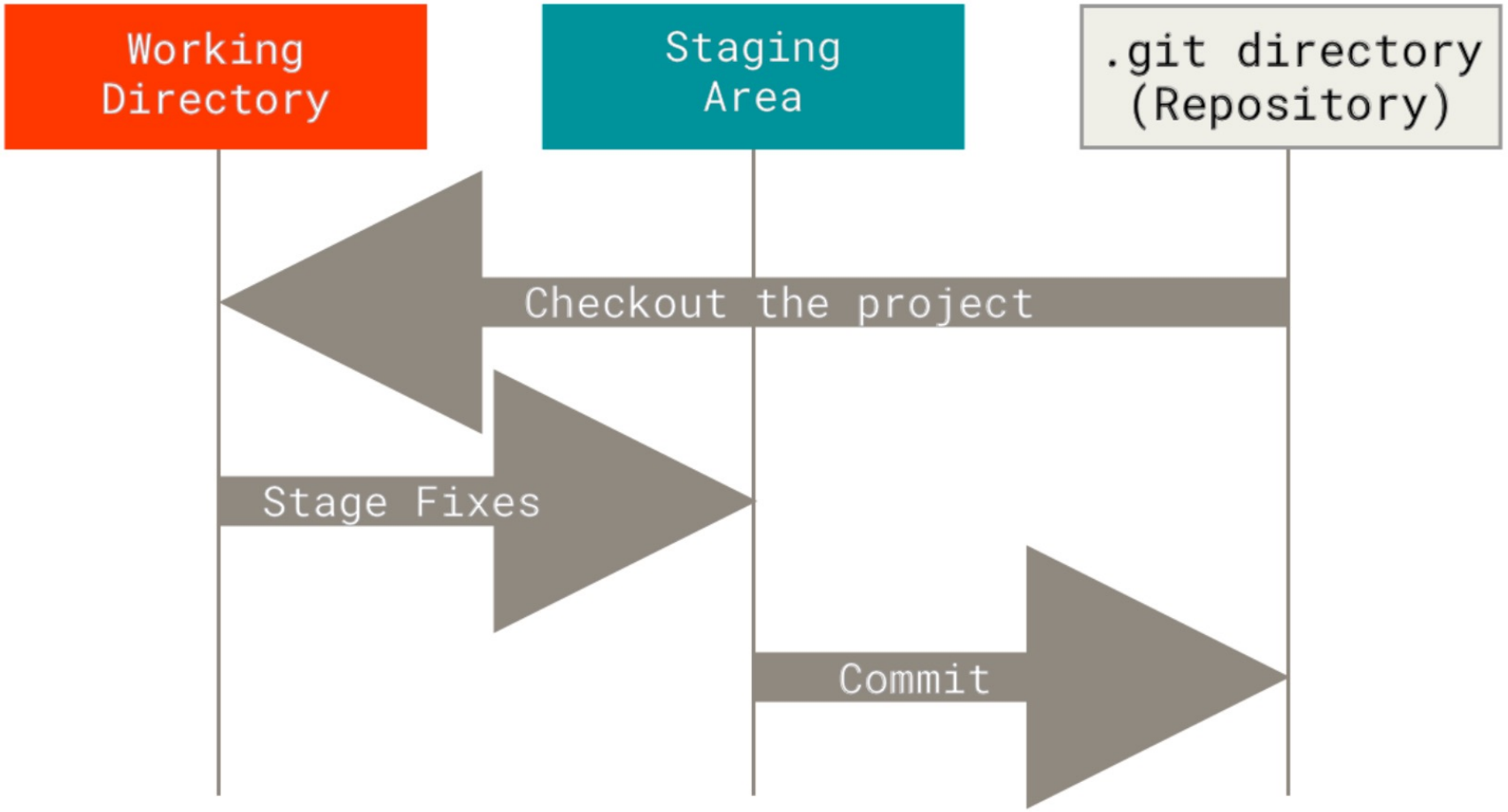
Pa rrezik

- Kur bëjmë veprime në Git, pothuajse të gjitha ato vetëm shtojnë të dhëna në bazën e të dhënave Git.
- Është e vështirë të detyrosh sistemin të bëjë diçka që nuk mund të zhbëhet ose ta detyrosh atë të fshijë të dhënat në çfardo mënyre.
- Ashtu si me çdo VCS, mund të humbim ose të ngatërrojmë ndryshimet që nuk i kemi ruajtur (commit), por pas kësaj është shumë e vështirë të humbim diqka, veçanërisht nëse bëjmë ruajtje të rregullta.
- Kjo e bën përdorimin e Git një kënaqësi sepse e dimë se mund të eksperimentojmë pa rrezikun e prishjes së rëndë të gjërave.

Tri gjendet

- Git ka tri gjendje kryesore në të cilat mund të qëndrojnë fajlat:
  - modified (të ndryshuar),
  - staged (të inskenuar) dhe
  - commited (të ruajtur)
- *Modified* do të thotë që ju keni ndryshuar fajlin, por nuk e keni ruajtur ende atë në bazën e të dhënave tuaja
- *Staged* do të thotë që ju keni shënjuar një fajl të modifikuar në versionin e tij aktual për të kaluar në ruajtjen e radhës.
- *Commited* do të thotë që të dhënat janë ruajtur mënyrë të sigurt në bazën e të dhënave tuaja lokale.
- Kjo na shpie në tre seksionet kryesore të një projekti Git:
  - The working tree (pema e punës),
  - The staging area (zona e skenës) dhe
  - Git directory (Git folderi ose direktoriumi).

Tri gjendjet



Tri gjendjet

- Pema e punës është një strukturë e një versioni të projektit.
  - Këta fajla nxirren nga baza e të dhënave e kompresuar në Git direktoriumin dhe vendosen në disk që t'i përdorni ose modifikoni.
- Zona e skenës është një fajl, i përfshirë në Git direktorium, që ruan informacione rreth asaj që do të përfshihet në ruajtjen e rradhës
  - Emri i saj teknik është "indeksi", por shprehja "zona e skenës" funksionon po aq mirë.
- Git direktoriumi është vendi ku Git ruan metadata dhe bazën e të dhënave të objekteve për projektin tuaj.
  - Kjo është pjesa më e rëndësishme e Git, dhe është ajo që kopjohet kur klononi një depo nga një kompjuter tjetër.

Rrjedha e punës

- Rrjedha e punës Git shkon kështu:
  1. Ju modifikoni skedarët në pemën tuaj të punës.
  2. Ju vendosni në mënyrë selektive vetëm ato ndryshime që dëshironi të bëhen në ruajtjen e rradhës, gjë që shton vetëm ato ndryshime në zonën e skenës.
  3. Ju bëni një (ruajtje) commit, i cili i merr fajlat ashtu siç janë në zonën e skenës dhe ruan atë shkrepje (snapshot) përgjithmonë në Git direktorium.
- Nëse një version i veçantë i një fajli është në Gid direktorium, ai konsiderohet *commited*.
- Nëse është ndryshuar dhe është shtuar në zonës e skenës, është *staged*.
- Nëse është ndryshuar por nuk është vënë në skenë, ai është *modified*.
- Në kapitullin e rradhës do të shtjellojmë në detaje këto gjendje.

Kush përdor Gitin

- Zhvillues nga hobistë deri te gjigantët masivë të teknologjisë si Facebook dhe Google.
- Bashkëpuntorë teknikë
- Qeveritë, për bashkëpunim në hartimin e ligjeve
- Shkencëtarët
- Shkrimtarët (jo të gjithë)
- Studentët (për punë në grup dhe tezë)

Git vs GitHub

- Git nuk është i njëjtë me Github.
  - Git është softueri i kontrollit të versionit që funksionon lokalisht në kompjuterin tuaj.
  - Nuk ka nevojë të regjistroheni për një llogari.
  - Nuk ka nevojë për internet ose Github për të përdorur Git.
- Github është një shërbim që ruan depot e Git në cloud.
  - E bën më të lehtë bashkëpunimin me njerëzit e tjerë.
  - Duhet të regjistroheni për një llogari për të përdorur Github.
  - Është një vend në internet për të ndarë punën që bëhet duke përdorur Git.

Përdorimi i Gitit

- Git u krijua si vegël e terminalit (CLI).
- Për ta përdorur atë, ne ekzekutojmë komanda të ndryshme Git në një Unix shell.
- CLI nuk është përvoja më miqësore për përdoruesit, por është në thelb të Git!
- Gjatë viteve të fundit, kompanitë kanë krijuar GUI për Git.
- Këto i lejojnë njerëzit të përdorin Git pa qenë nevoja të jenë ekspert të CLI.
- GUI-të e njohura Git përfshijnë:
  - Github Desktop
  - SourceTree
  - Tower
  - GitKraken
  - UnGit

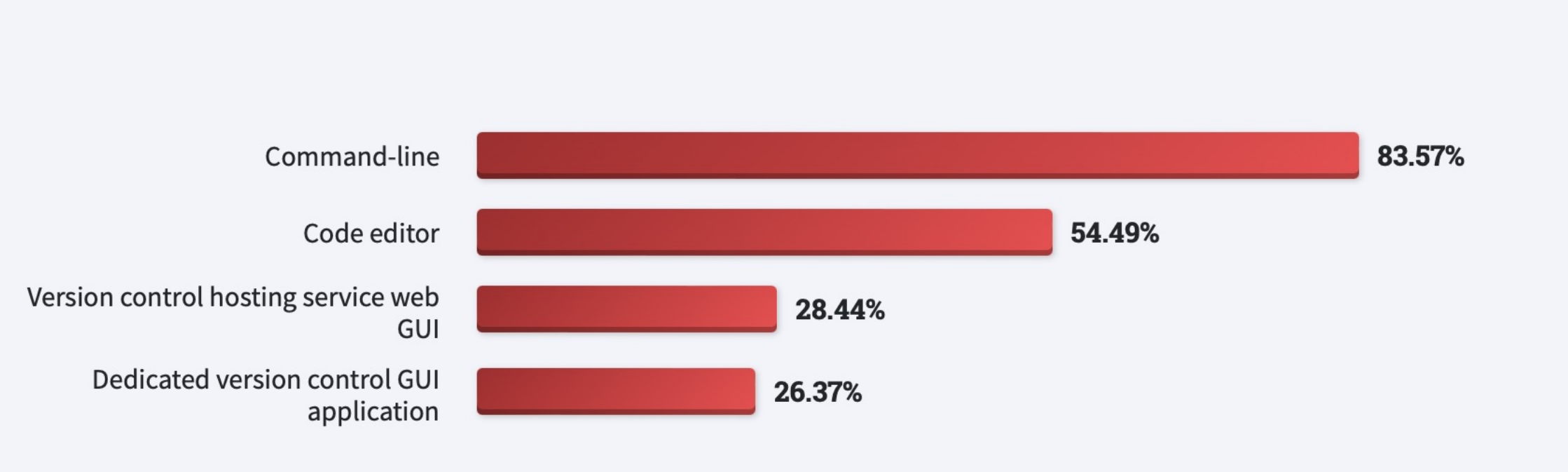
Klientët GUI

- Avantazhet
  - Më i lehtë për fillestarët në krahasim CLI
  - Më miqësor për t'u përdorur.
  - Mund të jetë një përvojë shumë më e mirë.
  - Disa njerëz preferojnë përvojën vizuale, madje edhe ata që mund të përdorin CLI
- Disavantazhet
  - Puna e brendshme të Git qëndron e fshehur nga përdoruesi
  - Shpesh çojnë në varësi nga një pjesë e caktuar e softuerit
  - Kur gjërat shkojnë keq seriozisht, mund të jetë shumë sfiduese për të rregulluar pa CLI
  - Ndërfaqet, butonat dhe menytë ndryshojnë ndërmjet GUI-ve të ndryshme

CLI

- Avantazhet
  - Git është një mjet CLI. I gjithë dokumentacioni dhe burimet në internet do t'i referohen CLIs
  - CLI mund të jetë shumë më e shpejtë sapo të ndiheni rehat me të
  - Disa nga veçoritë më të avancuara të Git janë të disponueshme vetëm në CLI
  - Komandat janë gjithmonë të njëjta, pavarësisht se në cilën makinë jeni!
- Disavantazhet
  - Nuk është aspak miqësore për fillestarët. Mund të jetë e vështirë për të mësuar dhe mbajtur mend komandat në fillim.
  - Edhe për disa përdorues të praktikuar, CLI nuk është thjesht një përvojë e mirë. Është çështje preference.

Interaksioni me Git



Instalimi në Windows – Git Bash

- Bash është CLI që përdoret gjerësisht nga zhvilluesit.
- Git u krijua për tu ekzekutuar në një ndërfaqe të bazuar në Unix (si Bash).
- Windows vjen me një CLI tjetër të quajtur Command Prompt që nuk është i bazuar në Unix.
- Paraqitet në skenë Git Bash!
  - Git Bash imiton përvojën e Git të bazuar në unix për makinat Windows
  - Shkarkoni Git për Windows (https://Git-scm.com)
- Gjeni fajlin .exe të shkarkuar për ta instaluar.
- Zgjidhni editor tuaj të preferuar gjatë fazës së instalimit.
- Më gjërësisht gjatë ushtrimeve në laborator.

Cila ndërfaqe?

- Në këtë lëndë, ne do të përdorim Git në CLI.
- Arsyeja, CLI është i vetmi vend ku mund të ekzekutoni të gjitha komandat Git
  - shumica e GUI-ve zbatojnë vetëm një nëngrup të pjesshëm të funksionalitetit Git për thjeshtësi.
- Nëse dini si të ekzekutoni CLI, lehtë mund të kuptoni se si të ekzekutoni versionin GUI, ndërsa e kundërta nuk është domosdoshmërisht e vërtetë.
- Gjithashtu, përderisa zgjedhja juaj e klientit GUI është një çështje e shijes personale, të gjithë përdoruesit do t'i kenë të instaluara dhe të disponueshme mjetet CLI.

Pyetje???